# Process Analysis Post Processing Tools

pyPA and PERMM

Barron H. Henderson

2013-01-30 to 2013-02-01

Each slide is shown above in an article format with extra notes shown here.

## Outline [Slide 2]

1. pyPA
   (a) Overview
   (b) Analysis Volumes
   (c) PA Mass Balance
   (d) Case-Studies
   (e) Hands-On
2. PERMM
   (a) Overview
   (b) Species Families
   (c) Net Reactions
   (d) Chemical Budgets
   (e) Chemical Indicators
   (f) Hands-On
3. PROJECT WORKSHOP

pyPA and PERMM are complimentary products that augment process analysis. Each section above is a large topic with many facets. Each section will be covered first in an overview manner followed by hands on applications. The applications will do a combination of getting you familiar with the concept and orienting you to the potential implications. There are three major sections: 1) a section on pyPA, 2) a section on PERMM, 3) a bring your own problem session. The first two sessions will be broken up into two half day sessions. The third section will have a half day to itself.

The first half day will be spent introducing pyPA, installing and verifying pyPA on your machines, conceptually covering analysis volumes, and spending some time creating simple process analysis volumes. We'll be covering a lot of fundamental concepts that will underpin much of the later steps. If you do not understand something, anything, ask for clarification. This is a pilot class and your input/feedback is critical.

# 1 Overview

## Introducing Process Analysis and Tools [Slide 3]

- Why do we need it?
- What is it? Process Analysis is PA
- What does it do?
- Where did it come from?

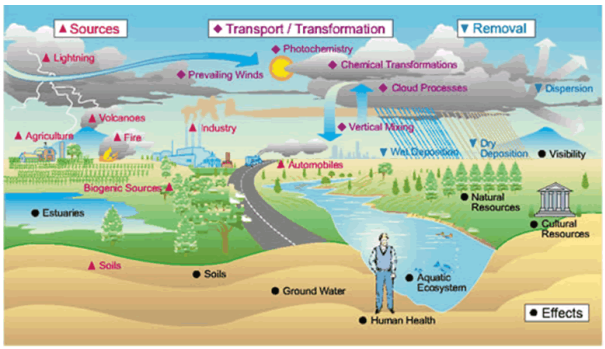
Process Analysis is a deceptively simple name, but analyzing processes leaves much implied. These following slides will describe precisely what process analysis is, why we need it, what it does, and where it came from. Understanding process analysis assumes a certain familiarity with modeling techniques that will also be described. Finally, this section will close with what process analysis can and cannot do for you. Process Analysis can solve many problems, but as with any tool it has limitations. As such, you must be able to choose when to (and not to) apply process analysis.

## Why do we need process analysis? [Slide 4]

- Higher-order models combine things we think we know in ways we might not understand

- Instrumenting higher-order models can help us understand our environment...
- Or the "data overwhelm" can obscure that information
- How can I use model instrumentation in a way that provides information instead of data?

## Conceptual model is inclusive [Slide 5]




- all physical and chemical processes
- by necessity excludes what we don't know

## Numerical Models "simplify" our world for computation [Slide 6]

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (\vec{u} c_i) = R_i(\vec{c}) + E_i - S_i$$

where

$c_i = c(\vec{x}, t)$ is the concentration of a chemical compound $(i)$ as it changes in time $(t)$ and space $(\vec{x})$

$\vec{u} = \vec{u}(\vec{x}, t)$ is the velocity vector in three dimensions $(u_x, u_y, u_z)$

$R_i(\vec{c})$ is the chemical production as a function of concentration of all chemical compounds $(\vec{c} = \{c\}_{i=1}^{\infty})$

$E_i$ is the emissions of the chemical compound

$S_i$ is sum of all loss processes

# Simplified models are still complex [Slide 7]

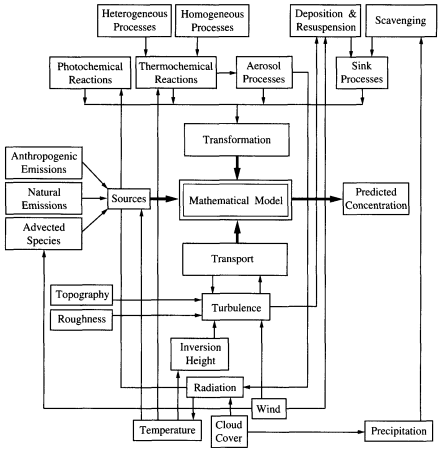



**FIGURE 25.1** Elements of a mathematical atmospheric chemical transport model.

- Emissions can be split into many categories
- Transport is the sum of advection, turbulent diffusion, etc
- Loss processes change from place to place
- It get's confusing fast

# Higher-order model [Slide 8]

- In fact, each input has its own model
- Chemical Transport Models ingest outputs from
  - Emission models, which ingest
    - activity models
    - emission factor models
  - Meteorology models, which ingest...
  - Light models, which ingest...
  - Chemistry models, which ingest...
  - Deposition models, which ingest...

## Higher-order model characteristics [Slide 9]

- Uncertainty is composed of
  - input uncertainties
  - science uncertainties
  - stochastic process uncertainties
- Each input model has uncertainties
- Combined uncertainty is difficult to quantify
  - due to mathematical assumptions for model solution
  - due to mathematical assumptions about error distributions

## Higher order model can surprise [Slide 10]

- Higher-order models can give unexpected results
- Atmospheric models are "open systems" that have "essentially unknowable" inputs[1]
- "Equifinality[2] is the principle that in open systems a given end state can be reached by many potential means." – Wikipedia.org 2008
- Process Analysis investigate of the system instead of the inputs and end state

In a 4-d modeling system that has the scope of an atmospheric model, it is unreasonable to think that we will have enough "knowns" to completely constrain the problem. Further, given the uncertainty in most model inputs and observations – it is possible to have an "over-constrained" deterministic result.

[1] Oreskes 1994 and later Beck 2002

[2] Ludwig von Bertalanffy's Preferred term

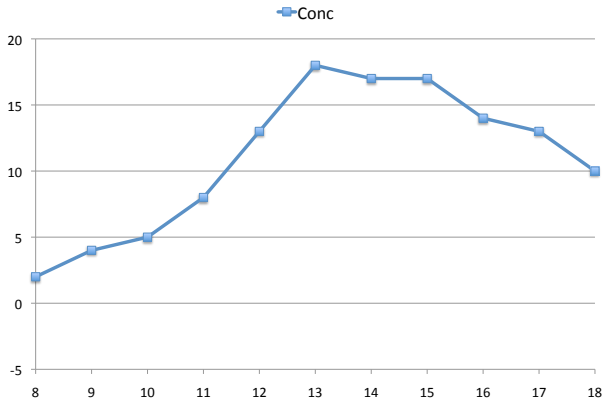
## Equifinality: failure or opportunity? [Slide 11]

- When a higher-order model gets the right (wrong) answer for a different reason than we expect:
  1. We can fix the model.
  2. We can investigate the short-comings of our conceptual model.
- We can use *model instrumentation*[3]:
  1. Extract information about the model solution
  2. Update/create a conceptual model from the model

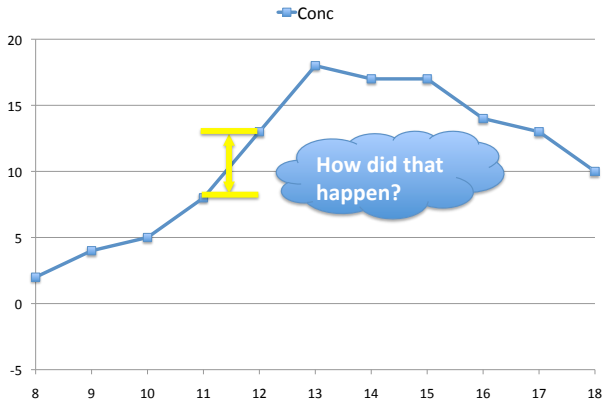
## PA details how predictions are made [Slide 12]

- Models Solve
  - How controllable processes (e.g. emissions) influence state variables (e.g. concentrations)?
  - How state variables influence uncontrollable processes (e.g. chemistry, advection, diffusion)?
  - How uncontrollable processes influence state variables? (e.g. PM2.5)
- PA outputs processes

## PA simply put [Slide 13]

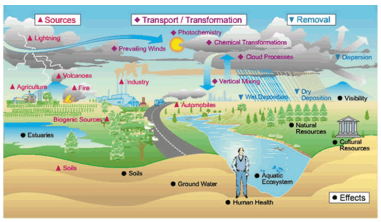



[3]model instrumentation is a term to describe taking samples of model inner workings




- Models tell you what changed.
- Process analysis tells you how it happened.

# Reconciles conceptual and numerical models [Slide 14]




=

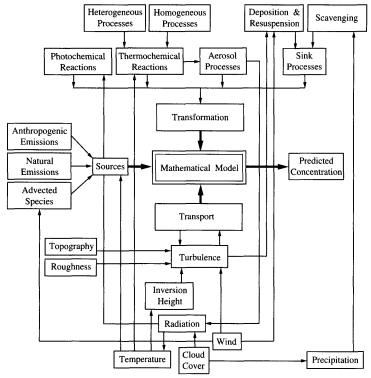



**FIGURE 25.1** Elements of a mathematical atmospheric chemical transport model.

# PA works for many models [Slide 15]

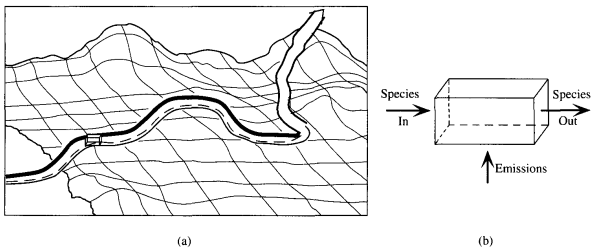



**FIGURE 25.2** Schematic depiction of (a) a Lagrangian model and (b) a Eulerian model.

- Process analysis can work for all types of models
  - Lagrangian
  - Eulerian
- Limited in scope only by the processes included

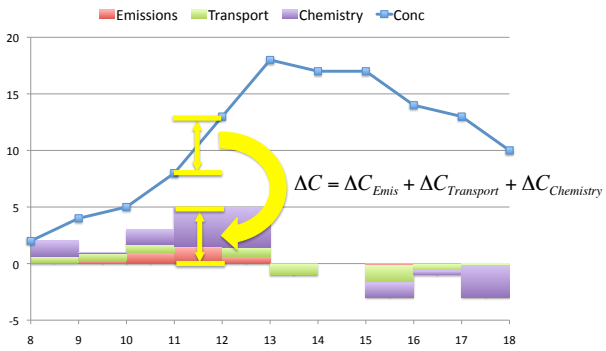
# Requires discrete solutions [Slide 16]

- Process Analysis code does require discrete solutions
- At a minimum time must be discretized
- Eulerian models have been used frequently and can be descretized in more dimensions

# Discrete time solutions [Slide 17]

- For a process to be quantified, it must have a distinct starting point and a distinct ending point
- These points do not need to be the same as the model simulation period
  - They can be a subset.
  - Can they be a super set? Only subjectively.
- Commonly applied for a subset period of interest
- Period of interest is usually broken down into sub-time steps for solution

# PA outputs model integration [Slide 18]




- for each discrete time, PA shows how much a process contributed to the observed change in concentration
- these can be summed across discrete time intervals if desired

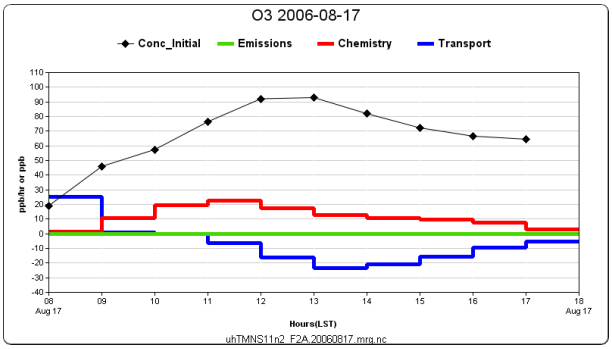
# Process analysis review [Slide 19]

- A review of what goes in to what comes out
  - terminology
  - technology
  - methodology

# Integrated processes rates [Slide 20]

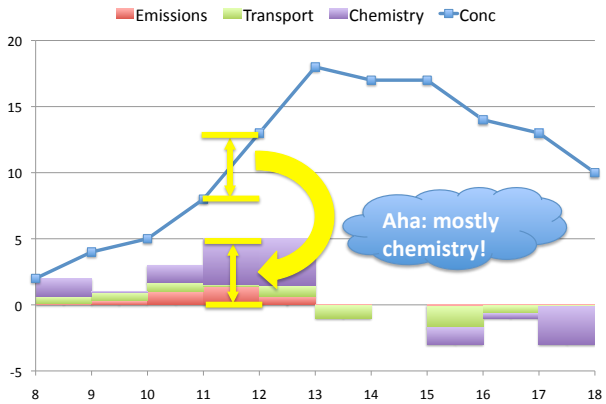
- the model is a numerical integrator
  - it takes in meteorology data
  - it takes in physical properties
  - it takes in emission rates
- the model converts its inputs into rates of change
- the model then integrates the change over a time
- an integrated rate is simply the change due to a single process

# Processes can offset each other [Slide 21]



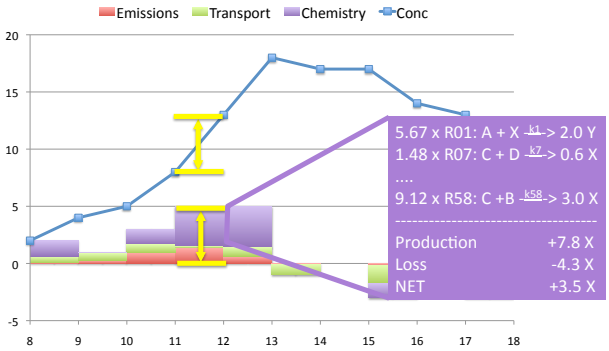

- Important to conceptualize processes as offsetting
- In this case, for example transport and chemistry have different signs depending on the time.

# Comparing process magnitudes [Slide 22]




By looking at process rates, we can identify which are important and which are not. This is not always immediately obvious from the concentrations.

# Chemistry is made of reactions [Slide 23]




Perhaps from this theoretical case, we could see that the production was abnormally high. Simple features spark interesting research!

# Integration of reaction rates [Slide 24]

- Chemistry is the sum of individual reactions
  - Reactions are solved using more complex representations
  - the partial differential equations are converted to ordinary differential equations
  - these are converted to Jacobians
- the values are input into a solver[4]
  - GEAR named for Gear
  - Livermore Solver for Ordinary Differential Equations Sparse (LSODES)
  - Rosenbrock (RODAS3 etc)
  - Euler Backward Iterative (EBI)
- Process analysis implementations must take special care to appropriately handle the solution

[4]Choice of solver may affect your results.

# Integrated reaction rates [Slide 25]

- Assuming the model correctly integrates the reaction rate
- It calculates many short time steps over which the reaction can be assumed to independent of other reactions

# Integrated reaction rates [Slide 26]

| Rxn # | Reaction | Flux (ppb/hr) | $NO_2$ | NO | O | $O_3$ | ... |
|---|---|---|---|---|---|---|---|
| 1 | $NO_2 + h\nu = NO + O$ | 127.13 | -127.13 | +127.13 | +127.13 | 0 | ... |
| 2 | $O + O_2 + M = O_3 + M$ | 151.76 | 0 | 0 | -151.76 | +151.76 | ... |
| 3 | $O_3 + NO = NO_2 + O_2$ | 129.76 | +129.76 | -129.76 | 0 | -129.76 | ... |

- Conceptually, you create a table of contributions
- The contributions are based on:
  - the total reacted mass of a first-order reactant
  - the stoichiometry of a product or reactant

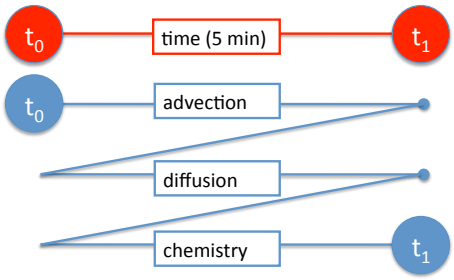
# Models are complex enough to be intractable [Slide 27]

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (\vec{u} c_i) = R_i(\vec{c}) + E_i - S_i$$

- In the last slide, I introduced the idea that the chemistry and physical processes are solved separately.
- In most cases, the solution as described above is inseparable.
- Chemical transport models break the model down in time to such a small size[5] that the error from separation should be minimized.
- This concept is called operator splitting and is fundamental to how process analysis works.

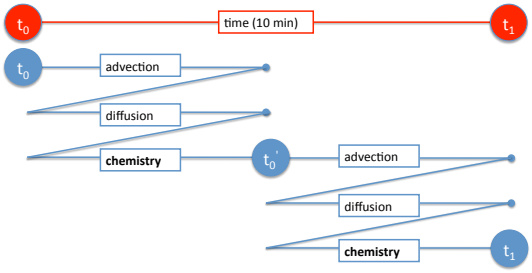
[5] Computational timestep is smaller than output timestep.

# Operator splitting [Slide 28]




- In operator splitting, you introduce a "fictitious time"
  - one process operates from $t_0$ to $t_1$ ($\Delta t$)
  - then, the next process operates from $t_0$ to $t_1$ ($\Delta t$)
  - then, the next process operates from $t_0$ to $t_1$ ($\Delta t$)
  - if $\Delta t$ was real time, it would be $t = t_0 + 3 \cdot \Delta t$
  - Instead after each process, it is $t_1'$, $t_1''$, and then $t_1$.
- The order affects availability of compounds.

# Operator splitting (continued) [Slide 29]




- Model output is usually output at 1-h or more
- Much longer than can be assumed with operator splitting
- Output timesteps made of many computational timesteps
- Time isn't the only discretization...

# Discrete spatial solutions [Slide 30]

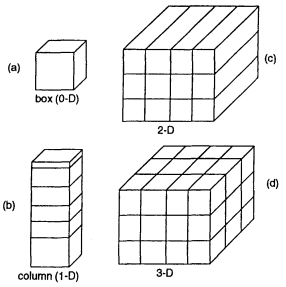



**FIGURE 25.3** Schematic depiction of (a) a box model (zero-dimensional), (b) a column model (one-dimensional), (c) a two-dimensional model, and (d) a three-dimensional model.

- Process analysis can be applied in any one of these discretizations
  - Python-based Environment for Reaction Mechanisms/Mathematics can be applied to all three
  - Python-based Process Analysis is really designed for b, c, and d

# Installation and Exercises [Slide 31]

- Example configurations of PA in CMAQ and in CAMx
- Step-by-Step installation of pyPA

# Model Configurations for PA [Slide 32]

- Chemical transport models are research tools
- Process analysis is a research tool
- As such, there is no single standard
- Examples shown here:
  - CAMx
  - CMAQ

The methods and options for generating Process Analysis output are model specific. We do our best to enable pyPA users to run PA on their host model, but the responsibility for documenting the process lies with the model user community. The instructions below are provided as a "best effort." Any discrepancies with the model's documentation should be assumed incorrect. Occasionally, we find errors in model documentation and report it to the model community. Anywhere our documentation should supersede the model documentation will be explicitly commented.

# CAMx Example Configuration [Slide 33]

- Create a CAMx run and execute it without PA (see Air Quality Models)
- modify the namelist to enable PA according to the CAMx manual (see inset below)

  , and
- rerun the CAMx simulation.

Below is an example code that can be added to a "standard" camx setup. This code is based on v4.51, but has been reused recently.

```
Probing_Tool = 'PA', ! (None,OSAT,GOAT,APCA,DDM,PA,RTRAC)          1
Number_of_Output_Species = 23,                                      2
Output_Species_Names(1) = 'NO',                                     3
Output_Species_Names(2) = 'NO2',                                    4
Output_Species_Names(3) = 'O3',                                     5
Output_Species_Names(4) = 'OLE',                                    6
Output_Species_Names(5) = 'PAN',                                    7
Output_Species_Names(6) = 'NXOY',                                   8
Output_Species_Names(7) = 'PAR',                                    9
Output_Species_Names(8) = 'TOL',                                    10
Output_Species_Names(9) = 'XYL',                                    11
Output_Species_Names(10) = 'FORM',                                  12
Output_Species_Names(11) = 'ALD2',                                  13
Output_Species_Names(12) = 'ETH',                                   14
Output_Species_Names(13) = 'CRES',                                  15
Output_Species_Names(14) = 'MGLY',                                  16
Output_Species_Names(15) = 'OPEN',                                  17
Output_Species_Names(16) = 'PNA',                                   18
```

```
Output_Species_Names(17) = 'CO',                                    19
Output_Species_Names(18) = 'HONO',                                  20
Output_Species_Names(19) = 'H2O2',                                  21
Output_Species_Names(20) = 'HNO3',                                  22
Output_Species_Names(21) = 'ISOP',                                  23
Output_Species_Names(22) = 'MEOH',                                  24
Output_Species_Names(23) = 'ETOH',                                  25
                                                                    26
Chemistry_Parameters = '$INPUT/CAMx4.5.chemparam.3',                27
                                                                    28
&PA_Control                                                         29
PA_File_Root = 'CAMx.${RUN}.200206${CAL}.PA',                       30
                                                                    31
Number_of_PA_Domains = 1,                                           32
Within_CAMx_Grid(1) = 1,                                            33
PA_Beg_I_Index(1) = 1,                                              34
PA_End_I_Index(1) = 4,                                              35
PA_Beg_J_Index(1) = 1,                                              36
PA_End_J_Index(1) = 4,                                              37
PA_Beg_K_Index(1) = 1,                                              38
PA_End_K_Index(1) = 4,                                              39
&                                                                   40
```

Notes:

- Line 1 would typically be set to 'None' instead of 'PA'
- Lines 2-25 define which species will have concentratons and now processes output
- Unlike CMAQ, CAMx cannot subset processes by species
- Unlike CMAQ, CAMx cannot create species families
- Lines 29-40 configure PA options and would not normally be present
- Line 30 defines the PA output file names
- Line 32 defines the number of subdomains
- Lines 33-39 define a specific subdomain

- Line 33: which grid it is in (1=Coarsest, 2=First nested)
- Lines 34-39 define the corners of the PA domain (1-based units) within the grid

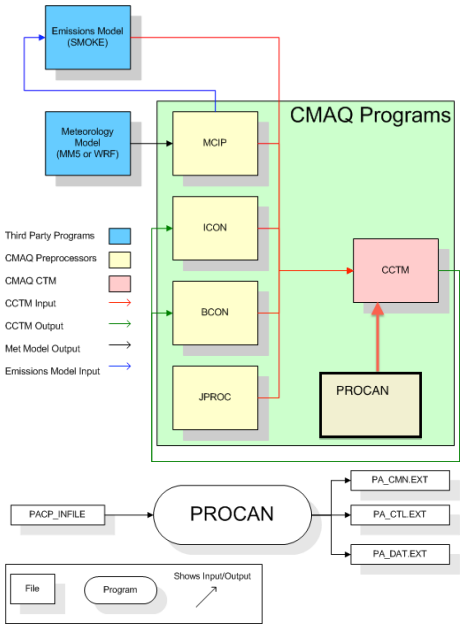
Exercise 1 : Modify the CAMx input to add a second PA domain, assuming Assume the outer grid is 240 cells west-east, 240 south-north, and 34 cells bottom-top. The current PA domain is in the south-west-bottom corner. Make the new PA domain in the north-east-top corner.

Answer:

Exercise 2 : Update the CAMX input assuming that nitrous acid was in the chemical mechanism as HNO2 instead of HONO and formaldehyde was in as CH2O instead of FORM.

Answer:

# CMAQ Configuration [Slide 34]



- Create a CMAQ run and execute it without PA[6],
- create a Process Analysis Control Language (PACL) input file,
- compile CMAQ's Process Analysis preprocessor (PROCAN)
- Run with new PACL file,
- recompile CMAQ executable with process analysis pointing to PROCAN output,
- modify the CMAQ execution script to enable PA,
- rerun the CMAQ simulation.

# CMAQ PACL [Slide 35]

- PACL is a rich language that allows for
  - defining species groups
  - defining complex relationships
  - subsetting species
  - subsetting processes by species
- It is quite elegant compared to CAMx's control system

For the rest of this class, we are going to focus on CMAQ. You all have been given data in a folder called 'PAPPT'. That folder contains a subfolder 'CMAQ'. 'CMAQ' contains 'v4.7', which contains 'data', 'scripts', and 'figs.' For the rest of the session, we'll assume that the "root" directory is 'v4.7', so if I say 'data' then I am refering to PAPPT/CMAQ/data.

Below is example Process Analysis Control Language (PACL) code that is provided as an input to CMAQ's Process Analysis preprocessor. The PACL file is typically housed in a "data/procan" directory in a file called 'pacp.inp'.

[6]Details available from Details available from CMAQ Operational Guidancehttp://www.cmaq-model.org/op_guidance_4.6/html

```
!=======================================================================
! Family Definitions
!=======================================================================
DEFINE FAMILY GNOY   = NO + NO2 + NO3 + 2*N2O5 + HONO +
                       HNO3 + PNA + PAN + NTR;

DEFINE FAMILY ASO4   = ASO4J + ASO4I;

DEFINE FAMILY ANO3   = ANO3J + ANO3I;

DEFINE FAMILY ANH4   = ANH4J + ANH4I;

DEFINE FAMILY AORGA  = AORGAJ + AORGAI;

DEFINE FAMILY AORGB  = AORGBJ + AORGBI;

DEFINE FAMILY AEC    = AECJ + AECI;

DEFINE FAMILY A25    = A25J + A25I;


!=======================================================================
! IPR_OUTPUTS
!=======================================================================

IPR_OUTPUT NO    = HADV + ZADV + HDIF + VDIF + EMIS +
                   DDEP + CLDS + CHEM          ;

IPR_OUTPUT NO2   = HADV + ZADV + HDIF + VDIF + EMIS +
                   DDEP + CLDS + CHEM          ;

IPR_OUTPUT O3    = HADV + ZADV + HDIF + VDIF +
                   DDEP + CLDS + CHEM          ;

IPR_OUTPUT CO    = HADV + ZADV + HDIF + VDIF +
                   EMIS + DDEP + CLDS + CHEM       ;

IPR_OUTPUT HNO3  = HADV + ZADV + HDIF + VDIF +
```

```
                        DDEP + CLDS + CHEM  + AERO;
IPR_OUTPUT PAN    = HADV + ZADV + HDIF + VDIF +
                    DDEP + CLDS + CHEM  + AERO;
IPR_OUTPUT H2O2   = HADV + ZADV + HDIF + VDIF +
                    DDEP + CLDS + CHEM        ;
IPR_OUTPUT GNOY   = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS + CHEM  + AERO;
IPR_OUTPUT FORM   = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS + CHEM        ;
IPR_OUTPUT SO2    = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS + CHEM        ;
IPR_OUTPUT SULF   = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS + CHEM  + AERO;
IPR_OUTPUT NH3    = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS +         AERO;
IPR_OUTPUT ASO4   = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS +         AERO;
IPR_OUTPUT ANO3   = HADV + ZADV + HDIF + VDIF + EMIS +
                    DDEP + CLDS +         AERO;
IPR_OUTPUT ANH4   = HADV + ZADV + HDIF + VDIF +
                    DDEP + CLDS +         AERO;
IPR_OUTPUT AORGA  = HADV + ZADV + HDIF + VDIF +
                    DDEP + CLDS +         AERO;
IPR_OUTPUT AORGB  = HADV + ZADV + HDIF + VDIF +
                    DDEP + CLDS +         AERO;
```

```
IPR_OUTPUT AEC     = HADV + ZADV + HDIF + VDIF + EMIS +                  77
                     DDEP + CLDS +              AERO;                  78
                                                                       79
IPR_OUTPUT A25     = HADV + ZADV + HDIF + VDIF + EMIS +                  80
                     DDEP + CLDS +              AERO;                  81
                                                                       82
                                                                       83
!=====================================================================  84
! IRR_OUTPUTS                                                          85
!=====================================================================  86
                                                                       87
IRRTYPE = FULL;                                                        88
                                                                       89
ENDPA;                                                                 90
```

Notes:

- DEFINE FAMILY creates a species group
- IPR_OUTPUT is followed by
  - a species or species group,
  - an equal sign (=) to separate group from processes
  - a + delimited list of processes
  - a semicolon (;)
- IRRTYPE = FULL outputs all reactions

Exercise 3 : Assume formaldehyde is enterred in the chemical mechanism as HCHO instead of FORM. Instead of updating the species name (e.g., in Exercise 2), create a species family that would make FORM correct.

Answer:

Exercise 4 : All compound specific line could be replaced by one line where the species name (e.g., NO) is replaced with ALL. The individual process names could be left out (e.g., leaving only the ;) to get all species. What benefits do you get from this level of control?

Answer:

After you create the pacp.inp file, you would compile and run procan (cd scripts/procan; ./bldit.pacp; ./run.pacp). After running procan, include files for CMAQ would be created in data/procan. At that point, you would recompile CMAQ to enable the use of these include files. Below is an example section of the CMAQ build script (csh) that is used to enable PA.

```
#> user choices: set process analysis linkages                          1
set PABase     = $GlobInc                                               2
set PAOpt      = pa_noop                                                3
set ICL_PA     = $PABase/$PAOpt                                         4
#> ICL_PA should point to the output from PROCAN                        5
echo "include SUBST_PACTL_ID $ICL_PA/PA_CTL.EXT;" >> $Cfile            6
echo "include SUBST_PACMN_ID $ICL_PA/PA_CMN.EXT;" >> $Cfile            7
echo "include SUBST_PADAT_ID $ICL_PA/PA_DAT.EXT;" >> $Cfile            8
```

Exercise 5 : Assume that the $M3HOME variable is set as the "root" CMAQ directory (i.e., contains scripts/procan, and data/procan) and that $M3HOME/data/procan contains the PROCAN output. Modify PABase and PAOpt so that ICL_PA points to your outputs.

Answer:

After re-compiling CMAQ, you need to configure the run script so that it outputs PA data. Below is an example modification of the CMAQ run script (csh) to set the PA domain.

```
set PA1file    = $EXEC"PA_1".$APPL  # CTM_IPR_1                        1
set PA2file    = $EXEC"PA_2".$APPL  # CTM_IPR_2                        2
set PA3file    = $EXEC"PA_3".$APPL  # CTM_IPR_3                        3
set IRR1file   = $EXEC"IRR_1".$APPL # CTM_IRR_1                       4
set IRR2file   = $EXEC"IRR_2".$APPL # CTM_IRR_2                       5
set IRR3file   = $EXEC"IRR_3".$APPL # CTM_IRR_3                       6
#> layer range for standard conc                                       7
setenv PA_BLEV_ELEV " 1 4"                                            8
```

```
setenv PA_BCOL_ECOL " 1 4"                                    9
setenv PA_BROW_EROW " 1 4"                                    10
```

Exercise 6 : CMAQ creates up to 3 files to hold all the PA data and 3 files to hold all the IRR data. Assume EXEC is set to "CMAQ" and APPL is set to "PRACTICE", what would the name of the first PA file be.

Answer:

Exercise 7 : As with exercise 2, create a new process analysis domain that is in the north-east-top corner. Answer:

## WRF-Chem [Slide 36]

- Create a WRF-Chem run and execute it without PA,
- modify the namelist to enable PA for the domain of interest, and
- rerun the WRF-chem simulation.
- Not a lot of options.

## Assuming you all just got model output... [Slide 37]

- What next?
- You just got a (or two) time-cube of data that you need to do something with.
- Model output units are typically intrinsic (per volume, per mole of air)

## Massive data with no obvious message [Slide 38]

## Air isn't really that discrete [Slide 39]

- The discretization in a Eulerian model creates an artificial boundary between our computational model and our conceptual model

  - Ambient air is a continuum
  - Plumes are continuums
- Enter Python-based Process Analysis

## Python-based Process Analysis (pyPA) [Slide 40]

- Python-based Process Analysis (pyPA) is a tool to help integrate model processes across space
- Conceptually
  - You have a problem that exists in a continuum
  - You break it into pieces to solve
  - You can't compare the processes from the pieces to your conceptual model
- Instead you must *analyze volumes* that are consistent with your conceptual model.

## Analysis volumes [Slide 41]

- This concept of analyzed volumes leads to the term "Analysis Volume" or AV for short
- Analysis volumes are simple a subset of grid cells that characterize a feature

## PA has a rich history [Slide 42]

- 1984: Jeffries and Tonneson compared how different chemical mechanisms arrive at different answers
- Available:
  - 1984 PKSS (Tonneson)
  - 1985: HR-RADM (Jang)
  - 1990ish: CAMx (Zion Wang)
  - 1997: CMAQ (Gibson)
  - 2001: Master Mechanism (NCAR)
  - 2007: WRF-Chem (Henderson)

## Why use pyPA? [Slide 43]

- Creating an analysis will require a processing tool
- Excel?
  - ad-hoc evaluation?
- C'mon, how about pyPA?
  - easy to use
  - free
  - reasonably well documented
  - plus, I like to help!

## Installing pyPA [Slide 44]

Known Platforms:

- Unix
- Mac
- Linux

Requirements:

- Python >=2.5
- NumPy >=1.2
- PyYAML
- netCDF4

## Requirements: first option [Slide 45]

Enthought Python Distribution

- free for academic use
- works on all platforms
- meets all package requirements

## Requirements: second option [Slide 46]

Ubuntu/Debian (similar for RedHat)

- sudo apt-get install python-dev
- sudo apt-get install python-setuptools
- sudo apt-get install python-virtualenv
- sudo apt-get install python-numpy
- sudo apt-get install python-scipy
- sudo pip install PyYaml
- download and install netCDF4 following their instructions

## Requirements: third option [Slide 47]

Not my machine... what can I do

- wget -nH –no-check-certificate https://raw.github.com/pypa/virtualenv/master/virtualenv.py
- python virtualenv.py class
- source ~/class/bin/activate
- pip install numpy
- pip install scipy
- pip install PyYAML
- download and install netCDF4 following their instructions (hopefully that machine you're on has netcdf installed)

## Requirements: fourth option [Slide 48]

1. Download virtualbox
2. Download my virtual machine
3. Run from there

At this point in the class, we will make sure that everyone has a working version of pyPA. Given the number of options above, we should be able to get there. I will have a USB drive with VirtualBox for Mac and Windows. I will also have a VirtualBox virtual machine that will work for anyone who is having too much trouble with their personal setup.

# 2 Analysis volumes

## Choosing an analysis volume is subjective [Slide 49]

- What is an analysis volume?
- In the examples, you should have all noticed a cubist theme.
- How many plumes are cubes?

## Single cell analysis volume [Slide 50]

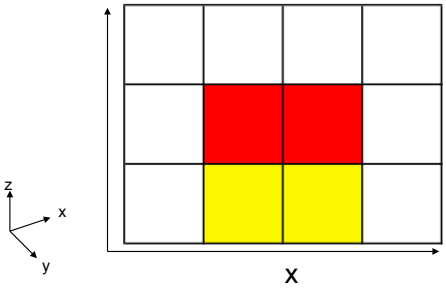
- Scale is important in choosing an analysis volume
- Assume for a moment a very coarse model
  - 1° latitude x 1° longitude[7]
  - first model layer includes the boundary layer
- In these conditions, the analyses of a single cell would fully characterize
  - my house
  - my neighborhood
  - my city
  - Rhode Island

[7] This is a reasonable resolution for a global model.

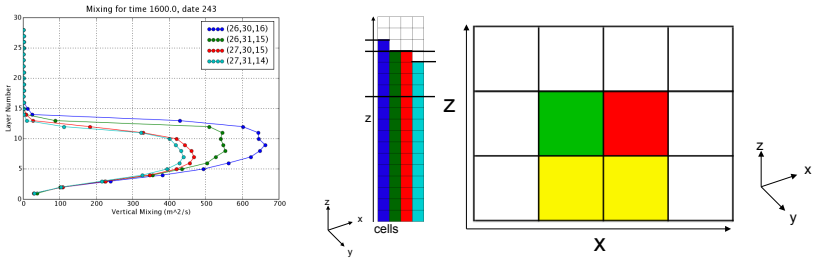
# Single cell analysis volume issues [Slide 51]

Assume instead that the vertical discretization has been improved, now the PBL is broken into 2 pieces. Layer 1 is in the PBL during the day and at night. Layer 2 is only in the PBL during the day.

- During the day, the PBL is well mixed within 1 hour
- Vertical advection
  - dominate in magnitude
  - provide no meaningful information
- Solution: a variable top analysis volume




# Variable top is variable in space [Slide 52]



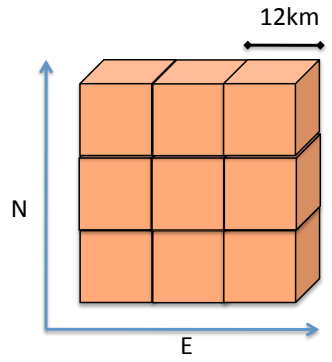

- Meta-processes: processes created by the redefinition of the analysis volume
  - entrainment (green)
  - detrainment (red)

# Variable top analysis volume issues [Slide 53]

Assume now, that the horizontal resolution has been better characterized to 12x12km[8]

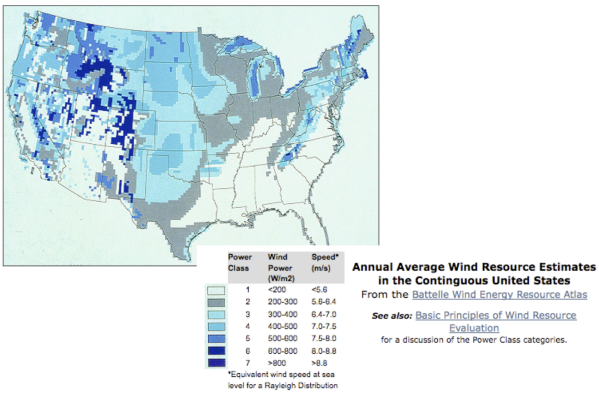
- Horizontal displacement of cell volume proportional:
  - wind speed
  - cell size
- Imagine for a moment a uniform $O_3$
  - constant moderate wind
  - 12 km x 12 km grid
- If v=0 m/s, for 100% turn over of the grid cell air in 1 hour u=? m/s

$3.3\overline{3}$m/s



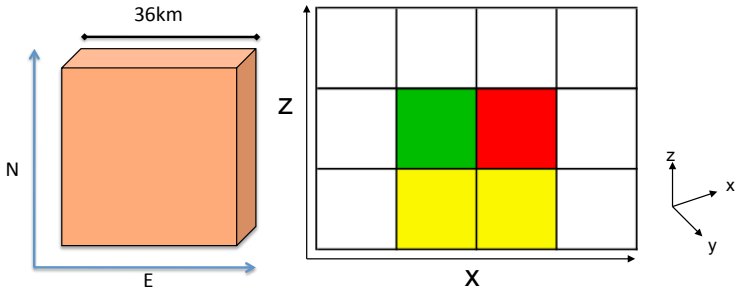

[8]Typical resolution for regional to continental scales.

## What types of wind are typical? [Slide 54]




- Most of US has $> 5.6$ m/s
- Southeast is between 0 and 5.6 /ms

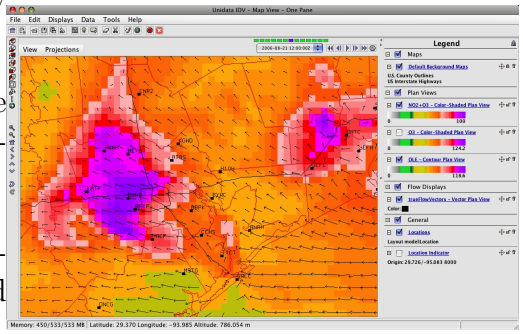
## Stationary analysis volume [Slide 55]




- Dilution *is* the solution
- Increasing the effective box size with variable top
- reduces the influence of transport
- requires a larger conceptual volume

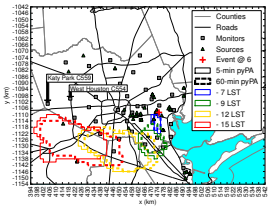
# Stationary analysis volume issues [Slide 56]

- Source and receptor may be very far apart
- The distance between the source and receptor will be proportional to dilution
  - mixes types of environments (e.g., near-road and rural)
  - averaging may hide key features
- Historically, models used a trajectory approach



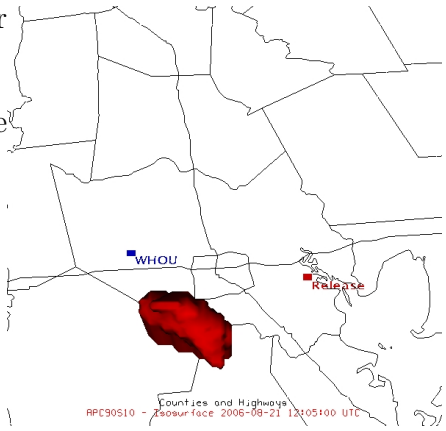
# Trajectory analysis volume [Slide 57]

- Consider the trajectory analysis volume as a moving stationary analysis
- Horizontal dimensions are constant
- (x,y) location change with time
- Useful for connecting sources with receptors in a well mixed PBL



## Trajectory analysis volume issues [Slide 58]

- Well mixed PBL assumption break down for strong sources
- Strong source can create plumes that have distinct vertical boundaries
- Plume analysis volumes




## Plume analysis volume [Slide 59]

- Plumes represent the most flexible case
  - Flexible in 4 dimensions
  - Limited only be the discretization of model time and space
- In some ways, the others are subsets of the plume analysis case

  **trajectory** plume whose bottom is connected to the surface

  **stationary** trajectory with 0 velocity

  **single cell** stationary with only one cell (variable top or not)
- Considering it this way will help to understand the techniques used to create them

## How is an analysis volume defined [Slide 60]

- Analysis volumes define a volume in 4-dimensions
  - The analysis volume must be defined by its location prior to the analysis and
  - The volume at each time in the analysis

- The prior time helps to connect multiple analyses if desired
- In most cases, the prior AV ($t_0$) will be set to the first time ($t_1$)
- For each time, the analysis volume is defined by on/off switches for each cell within the PA domain

## Creating analysis volumes [Slide 61]

- There are many ways to create analysis volumes
  - Easiest defines a single cell
  - Second easiest uses an assisted creation technique
  - Infinite number of flexible techniques
  - We'll discuss the most common plume identification techniques

## pyPA assistance [Slide 62]

- pyPA provides easy tools to create
  - single cell analysis volumes
  - stationary analysis volumes
  - Lagrangian analysis volumes
- To get started you need a minimal working copy of a pyPA configuration script

These next sections will gloss over a lot of the detail of the pyPA configuration script that will become important later. For the moment, be satisfied if following the instructions produces an analysis volume. This section will assume that you have successfully run CMAQ, though the approach is the same for CAMx. The *successful* simulation output files have been provided to you. These files will include integrated process rate files (PA_?), integrated reaction rate files (IRR_?), concentration files (CONC), and meteorological files. This section will also assume that you have successfully installed pyPA.

# Assisted analysis volume configuration [Slide 63]

Configuration pyPA for creating an analysis volume is relatively easy:

- pyPA will create a template configuration file for you to edit
- pyPA is designed to work with several models and needs your guidance to know which you have used

Exercise 8 : Produce a standard configuration file (test.yaml)

1. Invoke pyPA's main module (pyPA.main). Python invokes modules via the python −m flag. By default, the main The full command, is python −m pyPA .main. Below is an example execution.

```
Usage: main.py [-tq]                                                    1
                  [-i <init name>] [-f <final name>] <                  2
                     yamlfile>
                                                                        3
main.py: error: Requires a yaml file as an argument.                    4
   For a template use the -t option.  The template will
   be output to the stdout.
```

2. As with most programs, the −−help command provides more detailed help.

```
$ python -m pyPA.main --help                                            1
Usage: main.py [-tq]                                                    2
                  [-i <init name>] [-f <final name>] <                  3
                     yamlfile>
                                                                        4
Options:                                                                5
  -h, --help            show this help message and exit                 6
  -q, --qa              Check PA data                                   7
  -t, --template        Output template on standard out                 8
     (configurable with -m
                        and -c                                          9
  -m MODEL, --model=MODEL                                               10
                        Model can either be camx, cmaq                  11
                           or wrfchem (for use
                        with -t)                                        12
```

```
  -c MECHANISM, --mechanism=MECHANISM
                        Chemical mechanisms: cbivmech3,
                            cb05_ae4_aq,
                        cb05cl_ae5_aq, template (for use
                            with -t)
  -i INITIAL, --initial=INITIAL
  -f FINAL, --final=FINAL
```

3. Note the −t option that produces a template on the "standard output."

```
$ python -m pyPA.main -t
####### PA INPUT
#### Process Analysis uses model generated Integrated
    Process and Reaction
#### Rates (IPR/IRR) to create a conceptually well
    mixed volume that can
#### be analyzed to conceptually represent physical and
    chemical evolution.
####
#### There are 5 sections.  Commented options are
    optional.

##############################################################
####### OUTPUT PATH                                  #######
##############################################################
#### outfile is the path for results
outfile: <path for output file>

##############################################################
####### MODEL/MECHANISM IDENTIFICATION               #######
##############################################################
#### models supported are cmaq, camx, wrfchem
#### mechanisms supported cbivmech3
#model: cmaq|camx|wrfchem
#mechanism: cbivmech3|more-to-come

##############################################################
####### PA META WRAPPERS                             #######
##############################################################
```

```
#### Data that is needed by pyPA often has different
#### names, forms, and or meta data conventions.  pyPA
#### provides meta wrappers that will help you work with
#### specific models.  Please choose the wrapper that
    goes
#### with the model you are using.  If your model is not
#### provided, try pafile_master. pafile_master is the
    most
#### generic.
metawrapper: camx_pa_master|cmaq_pa_master|
    wrfchem_pa_master|pafile_master

##############################################################
####### INPUT FILES AND READERS                 #######
##############################################################
#### pyPA uses integrated reaction rate, integrated
#### process rate, instantaneous concentration data,
#### and meteorological data from files.  The variables
#### may be in any number of files with any number of
#### formats that each have their own readers.  Provide
#### the path for each file and the name of its reader
#### function.
####
#### READER FUNCTIONS BY MODEL
#### CAMx:
####  ipr: Integrated process rate file reader
####  irr: Integrated reaction rate file reader
####  vertical_diffusivity: vertical diffusivity (aka kv
    ) file reader
####  height_pressure: height/pressure (aka zp) file
    reader
#### CMAQ:
####  NetCDFFile: reads all CMAQ IRR, PA, and met data
#### WRF:
####  NetCDFFile: reads all WRF IRR, PA, and met data
#### OTHER:
####  Any 4-d (time/space) data can be used as long as
    you provide a reader
```

```
####  function for the data.  The reader function must    58
   return a NetCDFFile
####  object or a PseudoNetCDF object.  To specify your   59
   own reader put a
####  "from package_name import function_name" statement  60
    in place of a reader.
files:                                                    61
  - [<path to irrfile>, <reader>]                         62
                                                          63
##########################################################64
####### PA META VARIABLES                          #######65
##########################################################66
#### PA needs the name of variables to be used           67
#### for the following purposes:                         68
####  shape: name of PA volume shape variable            69
####         that is a 4-D array of booleans that        70
####         include or exclude cells in IRR/IPR         71
####  init_conc: Initial concentration                   72
####  final_conc: Final concentration                    73
shape: DEFAULT_SHAPE                                      74
init_conc: INIT                                           75
final_conc: FCONC                                         76
                                                          77
##########################################################78
####### 2-D MASK FOR HORIZONTAL SUBSETTING#######         79
##########################################################80
#### PA can subset your domain based on a text           81
#### file with 1s (include) and 0s (excluded)            82
####                                                     83
#### ascii_mask: path to text file.  If it does          84
####             not exist, a template is created        85
#ascii_mask: ascii_mask.txt                               86
                                                          87
##########################################################88
####### UNIT CONVERSION                            #######89
##########################################################90
#### unitconversion: dictionary of units to              91
####      convert.  Each unit to convert has             92
```

```
####       an expression that is evaluated to
####       convert the value to a new unit.
unitconversions:
    ppmV:
        expression: <value>*1000.
        new_unit: ppb
    ppm:
        expression: <value>*1000.
        new_unit: ppb

#############################################################
####### PA IRR/IPR VARIABES                          #######
#############################################################
#### species: list of species names
#### processes: list of process names
#### reactions: list of reaction variables or a
####     regular expression pattern for reaction
####     variable names
species: <list of species names>
processes: <list of process names>
reactions: '(IRR|RXN)_.*'
```

4. To capture the details, redirect the standard out to a file called temp.yaml: python −m pyPA.main −t > test.yaml

5. Open test.yaml in a text editor. You will edit this file to point it to the files provided from a CMAQ output.

6. Change outfile : <path for output file> to outfile : ../../ data/pappt/wholedomainshape .nc. When we run pyPA, it will now produce a file called wholedomainshape.nc

7. Change metawrapper: camx_pa_master|cmaq_pa_master|wrfchem_pa_master |pafile_master to metawrapper: cmaq_pa_master. Note that each metawrapper, except pafile_master is specific to a model. This allows pyPA to provide the most help it can. pafile_master is a generic wrapper if your model data comes from a model that is yet unsupported.

8. Copy − [<path to irrfile >, <reader>] once for each model output file you have been provided. For each model output file, replace <path to irrfile > with the path to the file. For all files, replace <reader> with NetCDFFile.

9. Replace species: <list of species names> with species: [O3]

10. Replace processes: <list of process names> with processes: [INIT]

11. Replace reactions: '(IRR|RXN)_.*'reactions: None

12. Compare your test.yaml with the one below.

```
###### PA INPUT                                                    1
#### Process Analysis uses model generated Integrated             2
    Process and Reaction
#### Rates (IPR/IRR) to create a conceputually well               3
    mixed volume that can
#### be analyzed to conceptually represent physical and           4
    chemical evolution.
####                                                              5
#### There are 5 sections.  Commented options are                 6
    optional.
                                                                  7
#############################################################     8
###### OUTPUT PATH                                      ######    9
#############################################################     10
#### outfile is the path for results                             11
outfile: ../../data/pappt/shape.nc                              12
                                                                  13
#############################################################     14
###### MODEL/MECHANISM IDENTIFICATION     ######                15
#############################################################     16
#### models supported are cmaq, camx, wrfchem                    17
#### mechanisms supported cbivmech3                              18
#model: cmaq|camx|wrfchem                                        19
#mechanism: cbivmech3|more-to-come                               20
                                                                  21
#############################################################     22
###### PA META WRAPPERS                                 ######    23
#############################################################     24
#### Data that is needed by pyPA often has different             25
#### names, forms, and or meta data conventions.  pyPA           26
#### provides meta wrappers that will help you work with         27
```

```
#### specific models.  Please choose the wrapper that goes
#### with the model you are using.  If your model is not
#### provided, try pafile_master. pafile_master is the most
#### generic.
metawrapper: cmaq_pa_master

##########################################################
####### INPUT FILES AND READERS              #############
##########################################################
#### pyPA uses integrated reaction rate, integrated
#### process rate, instantaneous concentration data,
#### and meteorological data from files.  The variables
#### may be in any number of files with any number of
#### formats that each have their own readers.  Provide
#### the path for each file and the name of its reader
#### function.
####
#### READER FUNCTIONS BY MODEL
#### CAMx:
####  ipr: Integrated process rate file reader
####  irr: Integrated reaction rate file reader
####  vertical_diffusivity: vertical diffusivity (aka kv) file reader
####  height_pressure: height/pressure (aka zp) file reader
#### CMAQ:
####  NetCDFFile: reads all CMAQ IRR, PA, and met data
#### WRF:
####  NetCDFFile: reads all WRF IRR, PA, and met data
#### OTHER:
####  Any 4-d (time/space) data can be used as long as you provide a reader
####  function for the data.  The reader function must return a NetCDFFile
####  object or a PseudoNetCDF object.  To specify your own reader put a
```

```
#### "from package_name import function_name" statement
    in place of a reader.
files:
  - [../../data/cctm/CCTM_e1a_Darwin9_i386_PA_1.ncf,
     NetCDFFile]
  - [../../data/cctm/CCTM_e1a_Darwin9_i386_PA_2.ncf,
     NetCDFFile]
  - [../../data/cctm/CCTM_e1a_Darwin9_i386_PA_3.ncf,
     NetCDFFile]
  - [../../data/cctm/CCTM_e1a_Darwin9_i386_IRR_1.ncf,
     NetCDFFile]
  - [../../data/cctm/CCTM_e1a_Darwin9_i386_CONC.ncf,
     NetCDFFile]
  - [../../data/mcip3/M_36_2001/METCRO2D_010722,
     NetCDFFile]
  - [../../data/mcip3/M_36_2001/METCRO3D_010722,
     NetCDFFile]

##############################################################
####### PA META VARIABLES                          #########
##############################################################
#### PA needs the name of variables to be used
#### for the following purposes:
####  shape: name of PA volume shape variable
####         that is a 4-D array of booleans that
####         include or exclude cells in IRR/IPR
####  init_conc: Initial concentration
####  final_conc: Final concentration
shape: DEFAULT_SHAPE
init_conc: INIT
final_conc: FCONC

##############################################################
####### 2-D MASK FOR HORIZONTAL SUBSETTING#####################
##############################################################
#### PA can subset your domain based on a text
#### file with 1s (include) and 0s (excluded)
####
```

```
#### ascii_mask: path to text file.  If it does
####              not exist, a template is created
#ascii_mask: ascii_mask.txt

##########################################################
###### UNIT CONVERSION                            ######
##########################################################
#### unitconversion: dictionary of units to
####       convert.  Each unit to convert has
####       an expression that is evaluated to
####       convert the value to a new unit.
unitconversions:
    ppmV:
        expression: <value>*1000.
        new_unit: ppb
    ppm:
        expression: <value>*1000.
        new_unit: ppb

##########################################################
###### PA IRR/IPR VARIABES                        ######
##########################################################
#### species: list of species names
#### processes: list of process names
#### reactions: list of reaction variables or a
####      regular expression pattern for reaction
####      variable names
species: [O3]
processes: [INIT]
reactions: None
```

# Assisted analysis volume [Slide 64]

- Hooray! You've just created a configuration file.
- We've created it to run a minimum pyPA analysis so that it will execute quickly
  - just one species

  - one process
  - no reaction

- If run, the output file will have IPR and IRR data
  - IPR includes CHEM and meta-processes
  - IRR includes nothing
- The output will also have the default analysis volume definition.
- Pretty easy so far, right?

## What is the default analysis volume [Slide 65]

- Horizontally:
  - the whole PA domain
  - possibly, but not necessarily the whole modeling domain
- Vertically
  - only within the PBL as defined by greater than floor, less than or equal to ceiling
  - PBL is model dependent
- PBL
  - CMAQ PBL height is defined by a variable in the meteorological input
  - CAMx has no PBL variable; PBL found using
    * algorithm from ENVIRON's vertavg utility (default)
    * algorithm the maximum elevation of non-default mixing

Exercise 9 : To create your first analysis volume, you'll need to run pyPA with the test.yaml file you created.

1. Now create your first analysis volume by running pyPA with the script.

```
$ python -m pyPA.main test.yaml
CMAQTransforms.py:217: UserWarning: Automatically
    offseting based on subdomain.
        tstart: 1
        tend: 24
        zstart: 1
        zend: 8
        xstart: 4
        xend: 36
        ystart: 6
        yend: 40
  \tyend: %d""" % (tslice.start+1, tslice.stop, kslice.
      start+1, kslice.stop, islice.start+1, islice.stop,
      jslice.start+1, jslice.stop))
DEFAULT_SHAPE DEFAULT_SHAPE
VOL VOL
VOL
1 1
AIRMOLS AIRMOLS
AIRMOLS
INIT_O3 INIT_O3
```

2. Create a script to view the average number of layers within the PBL.

```
# netCDF4 is a python-based reader for netcdf file
# it is written by Jeff Whitaker
from netCDF4 import Dataset;

# The Dataset object takes the path to a netcdf file
# and returns a file object
shapef = Dataset('../../data/pappt/shape.nc')

# NetCDF files all have a dimensions dictionary
# and a variables dictionary. Here we use the
# variables dictionary to grab the shape we made
shapev = shapef.variables['DEFAULT_SHAPE']

print 'DEFAULT_SHAPE'
print ' - Dimensions:', shapev.dimensions
```

```
print ' - Shape:', shapev.dimensions
print ' - Sum:', shapev[:].sum()

from matplotlib import use
use('TkAgg')
from pylab import *
title('Average Number of Layers within PBL')
pcolor(shapev[7:19].sum(1).mean(0))
colorbar()
savefig('../../figs/pappt/shape.png')
```

3. Run that script.

```
$ python shape.viewer.py ../../data/pappt/shape.nc
   DEFAULT_SHAPE
DEFAULT_SHAPE
 - Dimensions: (u'TSTEP_STAG', u'LAY', u'ROW', u'COL')
 - Shape: (u'TSTEP_STAG', u'LAY', u'ROW', u'COL')
 - Sum: 109425.0
0 .
1 .
2 .
3 .
4 .
5 .
6 .
7 .
8 .
9 .
10 .
11 .
12 .
13 .
14 .
15 .
16 .
17 .
18 .
19 .
```

```
20  .                                                    26
21  .                                                    27
22  .                                                    28
23  .                                                    29
24  .                                                    30
Average  .                                               31
```

4. Make a copy so you have a record of your progress. cp wholedomainshape.nc

## Assisted stationary analysis volume [Slide 66]

- Now we're going to use the output from the last section to create a new analysis volume.
- The new analysis volume will be stationary with a variable top, but will not cover the whole domain.
- We can exclude cells using an ASCII mask of 0s (off) and 1s (on)
- pyPA will even help you create the ASCII mask

Exercise 10 :

This is much easier, because much of the work was done in the assisted whole domain variable top.

1. Open test.yaml in a text editor (e.g., vi)
2. Change #ascii_mask: ascii_mask.txt to ascii_mask: ascii_mask.txt. This has the affect of "uncommenting" the ascii mask option.
3. Now rerun pyPA. Don't worry, it will look like some errors

```
$ python -m pyPA.main test.yaml                          1
CMAQTransforms.py:217: UserWarning: Automatically        2
    offseting based on subdomain.
        tstart: 1                                        3
        tend: 24                                         4
        zstart: 1                                        5
        zend: 8                                          6
        xstart: 4                                        7
```

```
        xend: 36
        ystart: 6
        yend: 40
  \tyend: %d""" % (tslice.start+1, tslice.stop, kslice.
     start+1, kslice.stop, islice.start+1, islice.stop,
     jslice.start+1, jslice.stop))
Traceback (most recent call last):
  File "/Library/Frameworks/EPD64.framework/Versions
     /7.3/lib/python2.7/runpy.py", line 162, in
     _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "/Library/Frameworks/EPD64.framework/Versions
     /7.3/lib/python2.7/runpy.py", line 72, in _run_code
    exec code in run_globals
  File "/Users/barronh/Development/pyPA/src/pyPA/main.py
     ", line 44, in <module>
    run()
  File "/Users/barronh/Development/pyPA/src/pyPA/main.py
     ", line 41, in run
    LoadPyPAFromYAML(yamlpath)
  File "/Users/barronh/Development/pyPA/src/pyPA/pappt/
     loader.py", line 90, in LoadPyPAFromYAML
    ext_mrg(job)
  File "/Users/barronh/Development/pyPA/src/pyPA/pappt/
     pappt.py", line 134, in ext_mrg
    Edit the template to include only those cells of
       interest"""
ValueError: File %s was not found; instead, a template
   was created.
    Edit the template to include only those cells of
       interest
```

4. The last two lines tell you to open the file and edit it. Open ascii_mask.txt in a text editor. Adjust the font so you can see the whole domain. It doesn't look like it yet, but that is a map of grid cells. Replace all the 1's with 0's.

5. Enable a 4x4 square starting at the bottom row and the 4th column. This will create a square in the southwestern corner (leaving three columns to the left

and 26 to the right all zeros).[9]

6. Change outfile: ../../data/pappt/wholedomainshape.nc to outfile: ../../data/pappt/partdomainshape.nc. When we run pyPA, it will now produce a file called partdomain.nc.

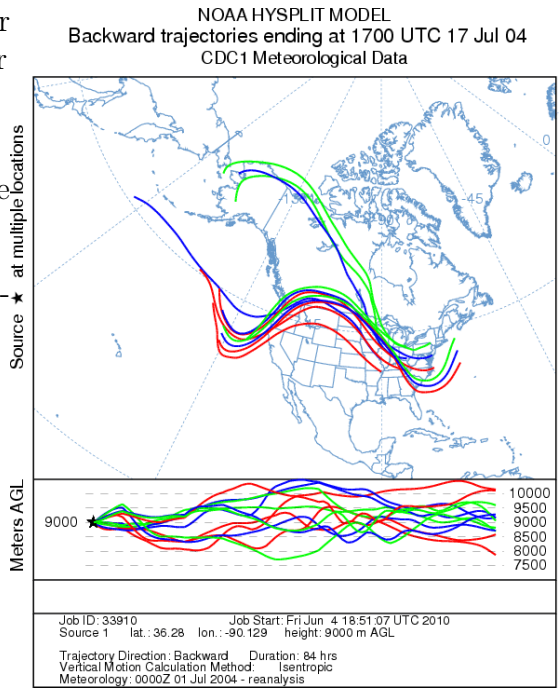
7. Rerun pyPA. This time, you'll get no errors. Compare your output to the listing below.

```
$ python -m pyPA.main test.yaml                                    1
CMAQTransforms.py:217: UserWarning: Automatically                  2
    offseting based on subdomain.
        tstart: 1                                                  3
        tend: 24                                                   4
        zstart: 1                                                  5
        zend: 8                                                    6
        xstart: 4                                                  7
        xend: 36                                                   8
        ystart: 6                                                  9
        yend: 40                                                   10
  \tyend: %d""" % (tslice.start+1, tslice.stop, kslice.            11
     start+1, kslice.stop, islice.start+1, islice.stop,
     jslice.start+1, jslice.stop))
DEFAULT_SHAPE DEFAULT_SHAPE                                        12
VOL VOL                                                            13
VOL                                                                14
1 1                                                                15
AIRMOLS AIRMOLS                                                    16
AIRMOLS                                                            17
INIT_O3 INIT_O3                                                    18
```

[9]For the practice we are creating a square, but there are no limitations on the shape. In general these shapes should be contiguous unless you have a specific reason to make it otherwise.

# Assisted trajectory analysis volume [Slide 67]

- Air from a source evolves as an air parcel is transported to a receptor
- This is a fairly common need.
- For example, this came up in the ROMANs project
- The trajectory could be defined using something like HYSPLIT




In practice, to start with you must decide where your trajectory analysis volume will go. A good approach would be to create a trajectory using HYSPLIT or STILT, for example, and then create a horizontal domain (perhaps variable in time) that would result in overlapping definitions. For our class example, we aren't going to use HYSPLIT. Instead, we will assume that at each hour the 4x4 grid made in the previous exercise moves north by 1 grid cell and east by one grid cell. Thus, the analysis volume at one time overlaps by more than 50% with the analysis volume at the second time.

1. Open your ascii_map.txt in a text editor again
2. Start at the northwest corner of your domain, change the (up 1, right 1) cell from 0 to 1
3. Repeat step one until you reach the eastern edge of the domain
   (a) Find and replace 0 1 0 0 0 0 0 0 with 0 1 1 1 1 1 1 1
4. Modify the rest by hand and compare your output to this listing.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3
0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4
0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5
0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6
0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7
1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 8
1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 9
1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 10
1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 11
1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 12
1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 13
1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 14
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 15
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 16
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 17
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 18
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 19
0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 20
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 21
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 22
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 23
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 24
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 25
0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 26
0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27
0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28
0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29
0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30
0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31
0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32
0 0 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 33
0 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34
0 0 0 0 0
0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35
0 0 0 0 0

5. Change outfile : ../../ data/pappt/partdomain.nc to outfile : ../../ data/pappt /lagrangianshape.nc. When we run pyPA, it will now produce a file called lagrangianshape.nc

6. Rerun pyPA. This time, you'll get no errors. Compare your output to the listing below.

```
$ python -m pyPA.main test.yaml
CMAQTransforms.py:217: UserWarning: Automatically
    offseting based on subdomain.
        tstart: 1
        tend: 24
        zstart: 1
        zend: 8
        xstart: 4
        xend: 36
        ystart: 6
        yend: 40
  \tyend: %d""" % (tslice.start+1, tslice.stop, kslice.
      start+1, kslice.stop, islice.start+1, islice.stop,
      jslice.start+1, jslice.stop))
DEFAULT_SHAPE DEFAULT_SHAPE
VOL VOL
VOL
1 1
AIRMOLS AIRMOLS
AIRMOLS
INIT_O3 INIT_O3
```

# Plume identification methodologies [Slide 68]

- plume analysis volumes can be very tricky
- defined by concentration gradient or effect gradient
  - inert tracers offer a way to define based on concentration gradient
  - reactive compounds (e.g., ethene, xylene) react away
- pyPA provides utilities pyPA.pappt.plumefinder
- plumefinder will identify plumes based on either
  - an explicit tracer
  - elevated concentrations (e.g. CO)

Exercise 11 : In this session, we are going to create a simple "plume-like" object by following CO. The CMAQ inputs that we have do not have a tracer inserted to follow a specific plume. Instead, all elevated CO areas will be identified.

1. Run python −m pyPA.pappt.plumefind
2. Review the help
3. Run python −m pyPA.pappt.plumefind test.yaml INIT_CO 0
4. mv massthresh.nc ../../data/pappt/
5. python shape.viewer.py ../../ data/pappt/massthresh.nc PLUME0pt90
6. Review the shape_PLUME0pt90*.png files
7. Note the outliers
8. python plumetrimmer.py ../../data/pappt/massthresh.nc PLUME0pt90
9. If there is enough time, review the above code to understand what was done.
10. python shape.viewer.py ../../ data/pappt/massthresh.nc PLUME0pt90
11. Review the shape_PLUME0pt90*.png files

# Requirements [Slide 69]

- shapes have only 2 requirements
    1. same dimensions as the PA output (typically 4-D)
    2. 0 means on, 1 means off
- you can make these files using any program you want
    - VERDI
    - IDV
    - NCL
    - anything...

# 3 PA Mass Balance

## PA integrates processes [Slide 70]

- Assuming you define a meaningful analysis volume...
- You have hundreds to thousands of individual time-cells
- Each one gives you a perfect account of itself
- pyPA integrates that data to give you a perfect account of the volume

## Process options by model [Slide 71]

- The processes available to include in the account depend on the model being run

  **CMAQ** INIT, XADV, YADV, ADJC, HADV, ZADV, HDIF, VDIF, EMIS, DDEP, CHEM, AERO, CLDS, PING, FCONC

  **CAMx** INIT, CHEM, EMIS, PTEMIS, PIG, A_W, A_E, A_S, A_N, A_B, A_T, DIL, D_W, D_E, D_S, D_N, D_B, D_T, DDEP, WDEP, INORGACHEM, ORGACHEM, AQACHEM, ACHEM, FCONC

## CAMx Processes [Slide 72]

**INIT:** Initial concentration

**CHEM:** Gas-phase chemistry

**EMIS:** Gridded emissions

**PTEMIS:** Point emissions

**PIG:** Plume-In-Grid

**DDEP:** Dry deposition

**WDEP:** Wet deposition

**A_?:** Advection from West (W), East (E), South (S), North (N), Bottom (B), Top (T)

**DIL:** Dilution

**D_?:** Diffusion from West (W), East (E), South (S), North (N), Bottom (B), Top (T)

**ACHEM:** CAMx v<=4.4 lumped aerosol chemistry

**?ACHEM:** aerosol chemistry - INORG = inorganic, ORG = organic, AQ = aqueous

**FCONC:** Final concentration

# CMAQ Processes [Slide 73]

**XADV:** Advection in the E-W direction (not compatible with YAMO advection)

**YADV:** Advection in the N-S direction (not compatible with YAMO advection)

**ZADV:** Vertical advection ADV3: Sum of XADV, YADV, and ZADV (not compatible with YAMO advection)

**ADJC:** Mass adjustment (not compatible with YAMO advection)

**HDIF:** Horizontal diffusion

**VDIF:** Vertical diffusion

**EMIS:** Emissions

**DDEP:** Dry deposition

**CHEM:** Chemistry

**AERO:** Aerosols

**CLDS:** Cloud processes and aqueous chemistry

**PING:** Plume-in-grid

## CMAQ Meta-processes [Slide 74]

**ADV2:** XADV + YADV (not compatible with YAMO advection)

**HADV:** Advection in the E-W-N-S direction (only compatible with YAMO advection)

**MADV:** Sum of HADV and ZADV (only compatible with YAMO advection)

**TADV:** Sum of XADV, YADV, ZADV, ADJC (not compatible with YAMO advection)

**TDIF:** Sum of HDIF and VDIF

**TRAN:** Sum of XADV, YADV, ZADV, ADJC, HDIF, and VDIF (not compatible with YAMO advection)

**TRNM:** Sum of HADV, ZADV, ADJC, HDIF, and VDIF (only compatible with YAMO advection)

## pyPA Process Configuration [Slide 75]

- processes: <list of process names>
- <list of process names> should be replaced by a subset of values from your models options
- The list format can follow two forms
  - two space indented lines starting with −
  - a list of words bounded by square braces [...]

## Typical configurations [Slide 76]

**CAMx:** processes: ['INIT', 'CHEM', 'EMIS', 'PTEMIS', 'PIG', 'A_W', 'A_E', 'A_S', 'A_N', 'A_B', 'A_T', 'DIL', 'D_W', 'D_E', 'D_S', 'D_N', 'D_B', 'D_T', 'DDEP', 'WDEP', 'AERCHEM', 'FCONC']

**CMAQ:** processes: [INIT, EMIS, HADV, ZADV, HDIF, VDIF, CHEM, AERO, CLDS, DDEP, FCONC]

- CAMx requires more processes because it does not provide many lumped categories (excluding AERCHEM)
- CMAQ has fewer processes because we are using lumped options
- Lumped options are common because, at a first order, we rarely distinguish between directions of transport

## Analysis volume considerations [Slide 77]

Don't forget that the importance of individual processes will change depending on how you define your analysis volume.

## Meta-processes [Slide 78]

- In addition to the model processes, pyPA adds meta-processes
- These processes are based on the change of the analysis volume

  **entrainment** mass added and volume increased

  **detrainment** mass removed and volume decreased

  **dilution** affect of volume changes are separated into a dilution term

- Processes happen in three dimensions

  1. horizontal plane is solved simultaneously
  2. vertical change follows

- The relative magnitude of these processes is sensitive to the order in which they are solved
- In most cases, these effects are small or compensating with transport (more to come later)

## Meta-process names [Slide 79]

**HDET** horizontally detrained (think moved out of)

**VDET** vertically detrained (think PBL collapse)

**HENT** horizontally entrained (think moved in to)

**VENT** vertically entrained (think PBL growth)

**EDHDIL** dilution from horizontal change

**EDVDIL** dilution from vertical change

Exercise 12 :
Modify your test.yaml to include the "typical" CMAQ configuration with the "default shape". Then create time-series plots in Excel (or your favorite typical graphing software).

1. Copy test.yaml to process.yaml.
2. Change the model output name ( outfile : ../../ data/pappt/wholedomainshape .nc to outfile : ../../ data/pappt/wholedomainprocess.nc) so that you don't overwrite you old results.
3. Comment out the ascii mask to use the whole domain (ascii_mask: ascii_mask .txt to #ascii_mask: ascii_mask.txt)
4. Change the process line from processes: [INIT] to processes: [INIT, EMIS, HADV, ZADV, HDIF, VDIF, AERO, CHEM, CLDS, DDEP, FCONC]
5. Re-run pyPA (look through your notes if necessary)
6. wholedomainprocess.nc now has AERO_O3, CHEM_O3, CLDS_O3, DDEP_O3, EMIS_O3, HADV_O3, TDIF_O3, ZADV_O3, INIT_O3, FCONC_O3, HENT_O3, HDET_O3, EDHDIL_O3, VENT_O3, VDET_O3, EDVDIL_O3
7. Dump these processes to a text file
   (a) ncdump −vAERO_O3,CHEM_O3,CLDS_O3,DDEP_O3,EMIS_O3,HADV_O3 ,TDIF_O3,ZADV_O3,INIT_O3,FCONC_O3,HENT_O3,HDET_O3,EDHDIL_O3 ,VENT_O3,VDET_O3,EDVDIL_O3 ../../data/pappt/wholedomainprocess .nc > ../../data/pappt/wholedomainprocess.txt
8. Review ../../data/pappt/wholedomainprocess.txt

# PERMM as a process plotter [Slide 80]

- PERMM's real strength is in the processing of chemical networks (more to come)
- It also, however, has good plotting functionality for pyPA process results
- The next section will be spent examining the effect of different analysis volumes on the plots we get

# Installing permm [Slide 81]

- Goto: permm.googlecode.com
- Download source
- unzip
- cd permm; python setup.py install

Exercise 13 : In this exercise, we are going to create a plot with permm.

1. python -m permm.main -g -c cb05_cmaq ../../data/wholedomainprocess.nc
2. select ozone from "reactants"
3. select "Process Plot" from "Execute"
4. Click go
5. Note that all processes are shown individually
6. Save the plot using the save button. Save it to ../../figs/wholedomainprocess.png

What are the two dominant processes? Are they meaningful?\newline
Answer:
\newline\newline\newline

## Examining effect of analysis volumes [Slide 82]

1. Install permm
2. Plot process.nc results with permm
3. Re-run process.yaml with various analysis volume definitions
   (a) part domain (partdomainshape.nc)
   (b) whole domain (wholedomainshape.nc)
   (c) trajectory (lagrangianshape.nc)
   (d) plume (massthresh.nc)

## Using analysis volumes we've made [Slide 83]

- simple!
- Add your shape file as an input under files. Add a line – [../../ data/pappt/ file.nc, NetCDFFile]
- Change shape: DEFAULT_SHAPE to appropriate name (e.g., shape: NAME
- where
  - file.nc is one of the files we made
  - and name depends on what the name of the variable is

Exercise 14 : In this exercise, we are going to produce pyPA outputs with processes for each domain. The "whole domain" is already done, because it is the default. For each of the others, you need to modify your yaml file to include the shape file and to use the right shape name.

1. Open test.yaml in a text editor
2. Change the model output name ( outfile : ../../ data/pappt/wholedomainprocess .nc to outfile : ../../ data/pappt/partdomainprocess.nc) so that you don't overwrite you old results.
3. Add an extra file line: – [../../ data/pappt/partdomainshape.nc below the last CMAQ input file line.

4. edit shape: DEFAULT_SHAPE to say shape: DEFAULT_SHAPE (no change for the first round)
5. Run pyPA python −m pyPA.main test.yaml
6. Repeat steps 1-4 with the options c and d
   (a) part domain (step2: partdomainshapeprocess.nc, step3: partdomainshape.nc, step 4: DEFAULT_SHAPE)
   (b) whole domain (step2: wholedomainprocess.nc step3: wholedomainshape.nc, step 4: DEFAULT_SHAPE)
   (c) trajectory (step2: lagrangianprocess.nc, step3: lagrangianshape.nc, step 4: DEFAULT_SHAPE)
   (d) plume (step3: massthreshprocess.nc step3: massthresh.nc, step 4: PLUME0pt90)

Exercise 15 : For each process file made in the last exercise, create a plot with permm.

1. python -m permm.main -g -c cb05_cmaq.yaml ../../data/*process.nc process-plot.py
2. Save the plot using the save button. Save it to ../../figs/*process.png
3. where * is wholedomain, partdomain, lagrangian or massthresh

\newline
What are the two dominant processes? Are they meaningful?
Answer:
\newline\newline\newline

# Importance of chemistry [Slide 84]

- You should find that the relative importance of the chemistry process
  1. is medium in the part domain example (part domain is near a source)
  2. gets diluted by the whole domain
  3. increases in the trajectory run
  4. increases with the plume definition

## Chemistry details [Slide 85]

- The integrated process rate for chemistry is conceptually made of hundreds of reactions
- Specifically CHEM_O3 = gross production - gross loss
- In Carbon Bond '05 for example
- Gross production
  - 2 radical-radical reactions: $C_nO_3 + HO_2 \longrightarrow 0 \cdot 2\,O_3 + ...$ where n = 2 and n > 2
  - 1 termolecular reaction: $O + O_2 \xrightarrow{M} O_3$
- Gross loss is from reactions with: radicals, olefins, terpenes, odd nitrogen, etc
- If loss of ozone by chemistry was a major issue, you'd want to know more

## What is in the integrated reaction rate? [Slide 86]

- Change of a reactant, assuming first order, due to a particular reaction
- $HCHO \xrightarrow{j} 2\,HO_2 + CO$
  - if IRR = 1/hr
  - 1 $CH_2O$ consumed per hour
  - 1 CO made per hour
  - 2 $HO_2$ made per hour

## Adding reactions to pyPA [Slide 87]

- Here we assume that CMAQ (or CAMx or WRF-Chem) Process Analysis had integrated reaction rates enabled
- There are several ways to handle this
- list them all individually
  - using same list format as discussed for processes

  - reactions are named by the host model (e.g., IRR_* where * is 1 to 3 digits)

- Use a regular expression

  - the default template uses a generic regular expression to find all reactions
  - '(IRR|RXN)_.*' where .* means any number of values

Exercise 16 : Assume that you know that the reactions of interest to you are chlorine and terpenoid reactions and that you know these are IRR_143, IRR_146, IRR_151, IRR_159.

1. Change the outfile to massthreshreactions.nc
2. Add just these reactions to the configuration file. Change reactions: None to reactions: [IRR_143, IRR_146, IRR_151, IRR_159]
3. Re-run pyPA and verify that you have these reactions using ncdump

Exercise 17 : Now try using the generic regular expression ('IRR_.*'). How many reactions are output? Keep this file at the ready.

# 4 Overview

## Processes are simpler than chemistry [Slide 88]

- As a first indicator, did you notice how many errors we got making a process plot?
- How many were process related?
- How many were reactions?

## Chemistry is ultra model specific [Slide 89]

- Pure chemistry isn't used by anyone
- Even MCM is near explicit, not explicit
- Most modeling exercises use fewer than 1000 reactions

- Some use less than 200
- Which are more right?

## Open Babel [Slide 90]

- There are whole softwares dedicated to translating between chemistry representations
- Unfortunately, they are still oriented more toward "pure" chemists than chemical modelers
- This year, at Atmospheric Chemical Mechanisms, this issue came up as something we need to address
- Regardless, chemical compounds and chemical reactions require prior information

## PERMM codifies that information [Slide 91]

- Not only does each model have it's own chemistry,
- It has its own format for storing that information
- The Kinetic Pre-Processor (KPP) is the closest thing to a model independent representation that can be used for models
- PERMM uses a KPP ish representation to store chemistry from many models

## Analysis independent definition [Slide 92]

- By storing the mechanisms in a database, PERMM allows for them to be reused.
- In the past,
  - analysis first required the codification of the chemical mechanism
  - this was usually done in a hard-coded manner within the analysis
  - it was extremely flexible
  - it was extremely error prone

## Reaction ordinals [Slide 93]

- Reactions are defined by both their reactants, products, and rate
- This leads to a rather dynamic definition that depends on the model representation
- This complexity doesn't work well with Fortran representation
- Instead, each reaction is assigned an ordinal
- Generic labels are not uncommon: BR07 might be the 7th Bromine reaction
  - CMAQ
  - KPP
  - SAPRC
- Ordinals are the most common

## Reactions have stoichiometry [Slide 94]

- The biggest problem for chemistry analysis is appropriate use of stoichiometry
- Each integrated reaction rate is based on the number of molecules a first order reaction would consume over some time
- Each reaction, however, can:
  - be first order, second order, third order
  - has products with non-unity stoichiometry
- Modeled chemistry also has fractional stoichiometry

## Fractional stoichiometry [Slide 95]

Fractional stoichiometry can come from several sources

1. Chemical lumping
2. Reaction lumping

## Chemical lumping [Slide 96]

**explicit** a compound is itself

**surrogate** a compound is assigned to one or more compound that represent its reactivity

**lumped** many compounds are combined into a "compound" with averaged properties

**structural** compounds are broken down into pieces, which are assigned to "compounds"

## Reaction analysis [Slide 97]

- A reaction analysis would normally require that:
  1. You know the reaction number (as implemented, not as documented)
  2. Know all the stoichiometries of all reactants and products
  3. Know what each product and reactant represent
- This is by its nature prone to errors
- Copy and pasting has led to errors in many projects

## Limits comparisons across mechanisms [Slide 98]

- The combination of *arbitrary decisions* makes analysis difficult
  - reaction identification
  - reaction lumping
  - species representation
- Comparing mechanisms using like assumptions is rare
- For example, how should prompt production of $HO_2$ be treated compared to secondary $HO_2$

# PERMM reduces barriers [Slide 99]

**permm** Python Environment for Reaction Mechanisms/Mathematics

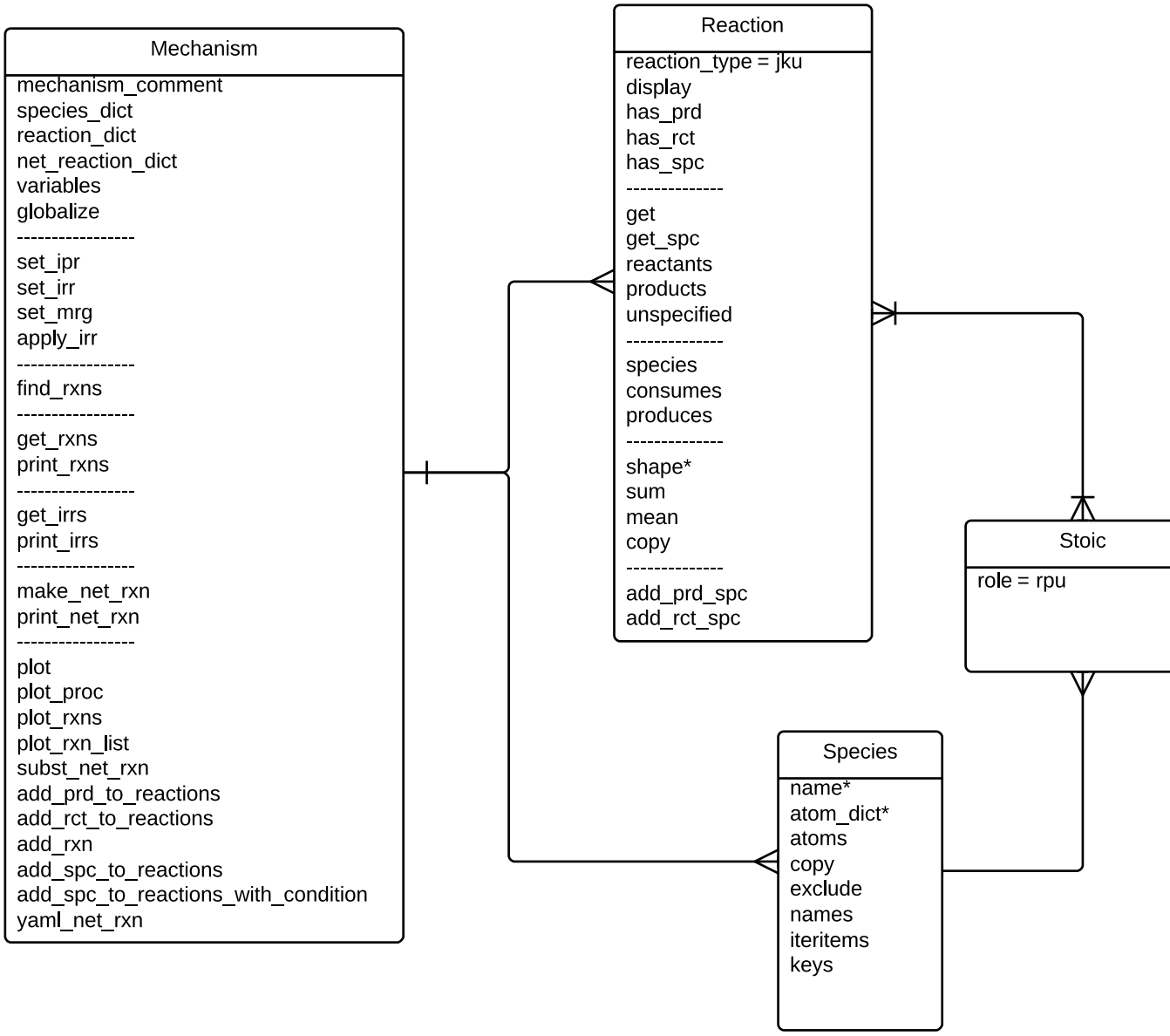
Goals:

1. Codify chemical mechanisms in a unified format
2. Store meta information about species
3. Provide access to objects using both the "standard" arbitrary ordinal and a query

Exercise 18 : Let's start by loading the massthreshreactions.nc file and making a pretty picture.

1. python −m permm.main −c cb05_cmaq −g ../../data/pappt/massthreshreactions .nc
2. Select O3 from the products
3. Try "Plot Reactions"
4. Repeat with Ox
5. Try "Print Net Reaction" with Ox
6. Do you think HNO4 production and loss is really important?
7. Try python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions .nc plot_ox.py

# PERMM Data Model [Slide 100]



## What is an environment? [Slide 101]

**program** a sequence of instructions written to perform a specified task

**environment** a type of computer software that assists computer programmers in developing software

**secret** permm is an environment that comes with basic analysis programs

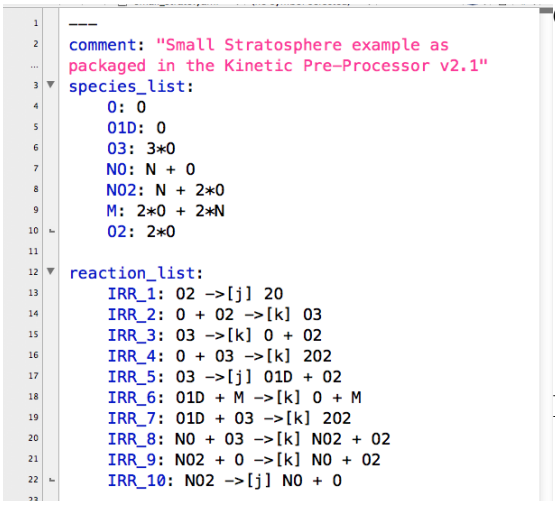
## What is a query? [Slide 102]

- Reactions are defined by their reactants, products, stoichiometry, and rates
- For example, take the reaction $CH_2O \xrightarrow{j} 2\,HO_2 + CO$
  - consumes formaldehyde
  - produces 2 hydroperoxy radicals and 1 carbon monoxide
  - rate is photolysis
- May have a different ordinal from model to model, but should be identifiable by its definition

## How is it loaded? [Slide 103]

- Each mechanism is specific to its implementation
- The mechanism is encoded once into a text file
- That text file is still largely built by hand
- There are tools to convert common formats (CMAQ mech.def; KPP *.eqn)

## What does it look like? [Slide 104]



```
---
comment: "Small Stratosphere example as
packaged in the Kinetic Pre-Processor v2.1"
species_list:
    O: O
    O1D: O
    O3: 3*O
    NO: N + O
    NO2: N + 2*O
    M: 2*O + 2*N
    O2: 2*O

reaction_list:
    IRR_1: O2 ->[j] 2O
    IRR_2: O + O2 ->[k] O3
    IRR_3: O3 ->[k] O + O2
    IRR_4: O + O3 ->[k] 2O2
    IRR_5: O3 ->[j] O1D + O2
    IRR_6: O1D + M ->[k] O + M
    IRR_7: O1D + O3 ->[k] 2O2
    IRR_8: NO + O3 ->[k] NO2 + O2
    IRR_9: NO2 + O ->[k] NO + O2
    IRR_10: NO2 ->[j] NO + O
```

Can have 5 sections:

- species_list
- reaction_list
- net_reaction_list
- species_group_list
- process_group_list

First two are most important!

## Can I make my own? [Slide 105]

I appreciate your enthusiasm! Yes, that is one of our exercises! Let's start by reviewing the objects.

# 5 Species families

## Species and families [Slide 106]

- All species are species families
- Remember the exercise where we made FORM = HCHO in CMAQ?
- Explicit species are species families with just one species
- Why bother? That means that you never treat anything differently.

# Examples of species [Slide 107]

```
species_list:
    CL: IGNORE
    NO3: 1N
    NO2: 1N
    'NO': 1N
    HONO: 1N
    HNO3: 1N
    PNA: 1N
    PAN: 1N + 2C
    PANX: 1N + 3C
    N2O5: 2N
    NTR: 1N + 2.5C
    PAR: 1C
    FORM: 1C
    MEOH: 1C
    ETHA: 2C
    ETOH: 2C
    ETH: 2C
    OLE: 2C
    ALD2: 2C
    ALDX: 2C
    IOLE: 4C
    ISOP: 5C
    TOL: 7C
    XYL: 8C
    TERP: 10C
```

```
species_group_list:
    - NOz=HNO3+PAN+PANX+NTR
    - NOx=NO+NO2
    - NOX=NOx+NO3+N2O5+HONO+PNA
    - NOy=NOX+NOz

    - NOzN=HNO3+PAN+PANX+NTR
    - NOxN=1*NOx
    - NOXN=NO+NO2+NO3+2*N2O5+HONO+PNA
    - NOyN=NOXN+NOzN

    - VOC=PAR + ETHA + MEOH + ETOH + ETH + OLE + IOLE + ISOP + TERP + FORM +
       ALD2 + ALDX + TOL + XYL
```

```
d0xr1bg1:perm barronh$ python -m perm -ic cb05_cmaq "print NO
print NO2
print NO + NO2
print NOy
print NOz
> print NOy-NOz
> "
ProcessWarning:
------------------------------------------------------
First argument was not a data file.
Attempting to continue with first argument as a script.
------------------------------------------------------
NO
NO2
NO + NO2 = 1.000*NO2 + 1.000*NO
NOy = 1.000*NTR + 1.000*N2O5 + 1.000*NO + 1.000*HONO + 1.000*PANX + 1.000*HNO3 + 1.000*PAN + 1.000*PNA + 1.000*NO2 + 1.000*NO3
NOz = 1.000*HNO3 + 1.000*PANX + 1.000*NTR + 1.000*PAN
NOy + xNOz = 1.000*N2O5 + 1.000*NO + 1.000*PNA + 1.000*HONO + 1.000*NO2 + 1.000*NO3
```

# Species properties [Slide 108]

- name* - the species family name
- atom_dict* - a dictionary of atoms by subspecies
- atoms - a method to return the species scaled by the number of atoms (e.g., N2O5.atoms('N') will return N2O5N = 2*N2O5
- names - list of subspecies
- iteritems - iterate over subspecies (like a dictionary)
- keys - subspecies list (like a dictionary)
- exclude - returns an exclude species... wait, what?

# What is an exclude species? [Slide 109]

- In most cases, we'll ask reactions about a species

- In some cases, it is extremely convenient to ask the mechanism for reactions that exclude a species.
  - NO.exclude() returns xNO

## Species operations [Slide 110]

- >>> NOx = NO + NO2
- >>> print NOx
- NO + NO2 = 1.000*NO2 + 1.000*NO
- >>> WHATAMI = NOx - NO2
- >>> print WHATAMI
- NO + NO2 + -(NO2) = 1.000*NO

Create a species that includes the following nitrogen compounds NO, NO2, NO3, N2O5, HNO3, NTR, HONO, PNA, PANX, PAN

1. python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions .nc
2. NOy = NO + NO2 + NO3 + N2O5 + HNO3 + NTR + PAN + PANX + HONO + PNA
3. Now find out how much NOy is consumed by IRR_1 print IRR_1[NOy]
4. What if you redefined NOy as everything except NO?

## Species format [Slide 111]

```
species_list:
    O: O
    O1D: O
    O3: 3*O
    NO: N + O
    NO2: N + 2*O
    M: 2*O + 2*N
    O2: 2*O
```

- Each species is defined by a algebraic equation of atoms.
- Only + and * are currently supported
- You can also leave this whole section blank

## Species family format [Slide 112]

```
species_group_list:
 - NOz=HNO3 + PAN + PANX + NTR
 - NOx=NO + NO2
 - NOX=NOx + NO3 + N2O5 + HONO + PNA
 - VOC=PAR + ETHA + MEOH + ETOH + ETH + OLE + IOLE
+ ISOP + TERP + FORM + ALD2 + ALDX + TOL + XYL
 - NMHC=NTR + ROOH + FORM + ALD2 + ALDX + PAR +
MEPX + MEOH + FACD + PAN + PACD + AACD + PANX + OLE
+ ETH + IOLE + TOL + CRES + OPEN + MGLY + XYL +
ISOP + ISPD + ETOH + ETHA + TERP
 - HC=NMHC + CH4
 - Radical=O1D + OH + HO2 + XO2 + XO2N + MEO2 +
HCO3 + C2O3 + CXO3 + ROR + TO2
 - RO2=XO2 + MEO2 + HCO3 + C2O3 + CXO3 + ROR + TO2
 - XO2_N=XO2 + XO2N
 - XO2_TO2=XO2 + TO2
 - XO2_N_TO2=XO2_N + TO2
 - RO2_NO2=RO2 + NO2
 - Ox=O3 + O + O1D + NO2 + 2*NO3 + 3*N2O5 + HNO3 +
2*PNA + PAN
```

- These can be pre-defined or defined on the fly
- The power of permm is largely tied to the definitions of species

# 6 Net reactions

## Reactions are made of stoichiometry [Slide 113]

- Stoichiometry tell you two things:
  - n-dimensional magnitude
  - role
- n-dimensional magnitude
  - no space or time: scalar ($CH_2O \xrightarrow{j} 2\,HO_2 + CO$)
  - time: 1-d vector (scalar times array of IRR)
  - time-space: 4-d (scalar times array from IRR)
  - time-space-ensemble: 5-d (scalar times array from IRR)
  - etc...

## Stoichiometry object [Slide 114]

- stoic is an overloaded ndarray
- the only real property it has is “role”

- 'r' is reactant
- 'p' is product
- 'u' is unknown, which will make more sense soon

# Reactions and net reactions [Slide 115]

- Reactions:
  - reaction_type = jku
  - display: make a pretty picture
  - has_spc: does the reaction have a species (has_prd: as a product; has_rct: as a reactant)
  - get, get_spc return the stoichiometry for a species
  - species, reactants, products, unspecified: list the species, reactants, products, or those that cannot be known
  - consumes, produces: provide an array that tells how much of a species is consumed or produced (even if it is not in the reactions)
  - shape* sum mean copy

Exercise 19 : Add reactions IRR_1, IRR_2, and IRR_3 together to create your first "net reaction."

1. python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions.nc
2. NR1 = IRR_1 + IRR_2 + IRR_3
3. print NR1
4. NR1.display(8)
5. NR1.display(0)
6. print NR1.sum()
7. print NR1.sum().display(8)
8. print NR1[NOy.atoms('N')]

## One mech to rule them all [Slide 116]

mechanism_comment

species_dict

reaction_dict

net_reaction_dict

variables

globalize

## Mechanism user methods [Slide 117]

- find_rxns - core query mechanism; finds reactions based on their properties and returns a list of names
- get_rxns - instead of returning names, returns objects
- print_rxns - doesn't return anything, just prints the reactions
- get_irrs - same as get_rxns, but returns reactions with IRR applied
- print_irrs - same as print_rxns, but returns reactions with IRR applied

## Query syntax [Slide 118]

- all functions use the same properties as find_rxns
- for more information on any function, type help (...) where ... is the function name.

```
Help on method find_rxns in module permm.Mechanism:

find_rxns(self, reactants=[], products=[], logical_and=True, reaction_type=None) method
of permm.Mechanism.Mechanism instance
    Get reaction names that meet the criteria specified by reactants, products and logic
al_and.

    reactants - filter mechanism reactions for reactions with all reactant(s)
    products - filter mechanism reactions for reactions with all product(s)
    logical_and - a boolean indicating how to combine reactant and product filters
                  True: reaction is in both filters (i.e. reactants AND products)
                  False: reaction is in either filter (i.e. reactants OR products)
```

Exercise 20 : Let's test out find_rxns and then try the same command in get_rxns, get_irrs, and print_irrs.

1. python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions .nc
2. Try both find_rxns(NO2) or find_rxns([NO2]); species and species lists are indistinguishable to find_rxns.
3. Try find_rxns(NO2, reaction_type = 'j'); reaction_type limits the reactions to photolysis (j) or (k) or unknown.
4. Try both find_rxns(NO2, NO) or find_rxns(NO2, [NO]); if both reactants and products are specified, both must be in the reaction.
5. Try find_rxns(NO2, NO, logical_and = False); if both reactants and products are specified, but logical_and is false than one or the other must be in the reaction.
6. Try find_rxns(NO2 + NO); species are indistinguishable from species groups.
7. Try find_rxns(NO2 + NO, HNO3); this gives NOx going to HNO3.
8. Try find_rxns(NO2 + NO, −HNO3); this gives NOx going to anything except HNO3.
9. Try some of the same commands with get_rxns, then with plot_irrs.

# Net reaction utilities [Slide 119]

- make_net_rxn - creates a net reaction from a query
- print_net_rxn - prints a net reaction from a query
- subst_net_rxn - replaces reactions with their net reaction

Exercise 21 : In this we will try creating the same net reaction as before (IRR_1 + IRR_2 + IRR_3) from a query.

1. First, show the original results print (IRR_1 + IRR_2 + IRR_3).sum()
2. Start simple. print_rxns(O+NO+NO2, O+NO+NO2)

(a) exclude radicals (i.e. -Radical)

(b) exclude NO3

(c) exclude CLO

3. print_rxns(O+NO+NO2, O+NO+NO2)

4. It isn't perfect, but are you sure you want to exclude these reactions? Compare to the original.

5. print (IRR_1 + IRR_2 + IRR_3).sum() and then make_net_rxn([O+NO+NO2,−NO3,−Radical,−CLO],O+O3+NO2).sum()

# Plotting commands [Slide 120]

- plot - plot a generic array of the time dimension
- plot_proc - easy access to reproduce the process plots
- plot_rxns - a query method of producing reaction plots
- plot_rxn_list - a method for producing reaction plots from a list of reactions

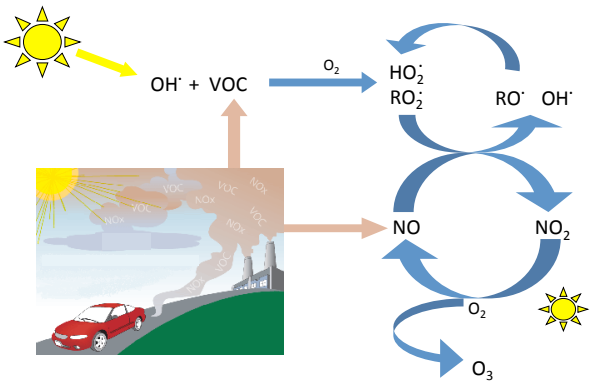
Exercise 22 : Explore plot_rxns using the help function. It is very similar to find_rxns, but has some bonus features.

1. When you understand it, create a plot based on your own query (or the one we developed above) and plot_rxns.

2. When you're done, recreate the plot by combining get_irrs and plot_rxn_list

# 7 Chemical Budgets

## Production/destruction or cycles [Slide 121]




## Initiation, propagation, and termination [Slide 122]

- Radical cycles (or NOx) can be defined by three terms
  - Initiation is a reaction that has no radical reactants, but has radical products
  - propagation has a radical reactant and a radical product
  - termination has a radical reactant but no radical product

Exercise 23 : How-to calculate initiation using permm

1. python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions .nc
2. rad_init = make_net_rxn(−Radical, Radical)
3. print rad_init.sum()
4. Note that at this point we're seeing PNA as a major source even though PNA is produced and consumed rapidly. The next lines replace regular reactions with net reactions for the duration of our session.

5. subst_net_rxn(PNA, PNA, logical_and = False)
6. subst_net_rxn(PAN, PAN, logical_and = False)
7. subst_net_rxn(PANX, PANX, logical_and = False)
8. subst_net_rxn(HONO, HONO, logical_and = False)
9. make_net_rxn(−Radical, Radical).sum()
10. How has the initiation changed?
11. make_net_rxn(−Radical, Radical)[6:].sum()
12. How did it change again when we time subset it?

Exercise 24 : How-to calculate propagation using permm

1. python −m permm.main −c cb05_cmaq −i ../../data/pappt/massthreshreactions .nc
2. rad_prop = make_net_rxn(Radical, Radical)
3. print rad_prop.sum()
4. Note that at this point we're seeing O1D as a propagator. We often consider this an initiation.The next lines replace regular reactions with net reactions for the duration of our session.
5. subst_net_rxn(O1D, O1D, logical_and = False)
6. make_net_rxn(Radical, Radical).sum()
7. How has the propagation changed?

# 8 Chemical indicators

# Chemical indicators [Slide 123]

- Chemical indicators tell us something about the system
- Is it sensitive to NOx or VOC, for example.

## Common chemical indicators [Slide 124]

1. P(O3)
2. P(O3)/P(NOz)
3. P(H2O2)/P(HNO3) - Sillman ratio

## Sillman ratio [Slide 125]

- A combination of theory and pragmatism
  - Note to self - show derivation

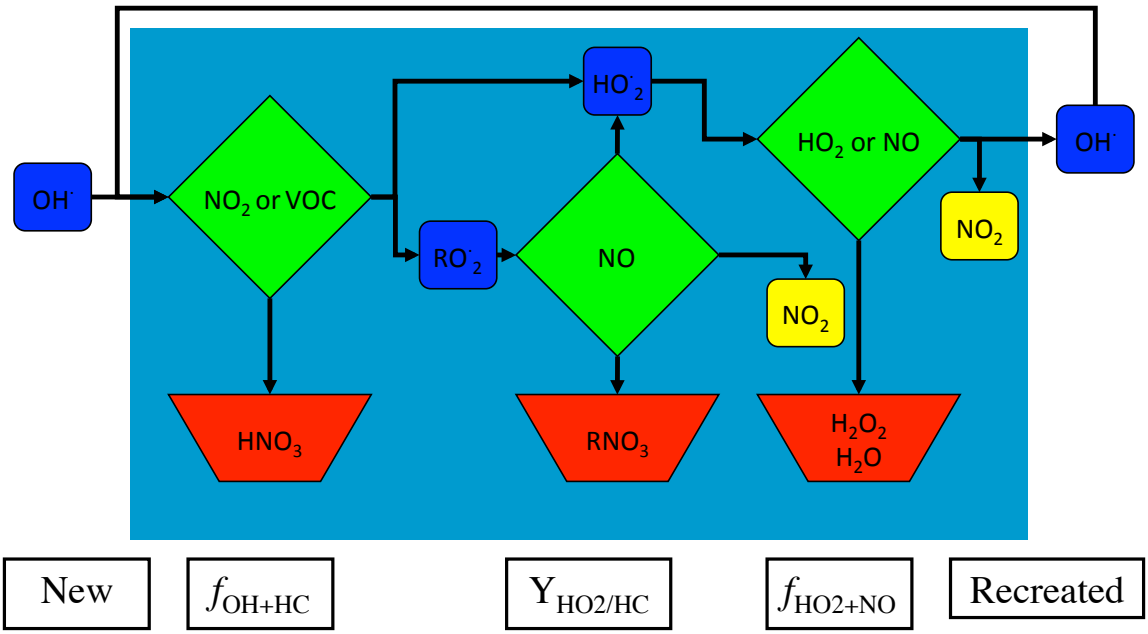
Exercise 25 : How-to: Sillman ratio

1. sillman = make_net_rxn([], H2O2)[Radical]/make_net_rxn([], HNO3)[Radical ]
2. plot(sillman)
3. import pylab; pylab.show()

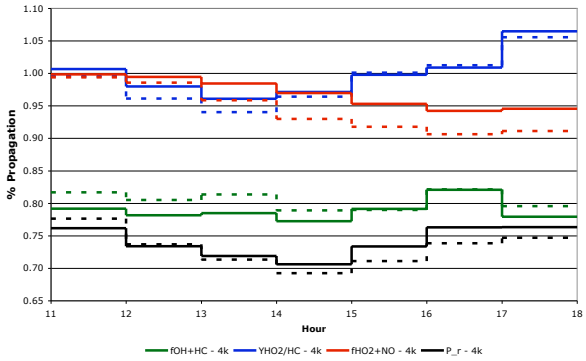
\newline
Exercise 26 : How-to: Augmented Sillman ratio

1. import pylab; pylab.interactive(True)
2. sillman = make_net_rxn(Radical, −Radical−NOz)[Radical]/make_net_rxn( Radical, [−Radical, NOz])[Radical]
3. plot(sillman)
4. Not right
5. Make some net reactions and try again.

# Propagation in detail [Slide 126]




# Example results [Slide 127]




# How-to: Arnold et al. [Slide 128]

x

# 9 Hands-on

## Let's go! [Slide 129]

x